EK-NIA20-RM-001

NIA20 Technical Reference Manual

digital™

EK-NIA20-RM-001

NIA20 Technical Reference Manual

Prepared by Educational Services
of
Digital Equipment Corporation

1st Edition, January 1985

The material in this manual is for informational purposes and is subject to change without notice.

Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual.

This book was produced on a DIGITAL Word Processing System. Book production was done by Educational Services Development and Publishing in Marlboro, MA.

The following are trademarks of Digital Equipment Corporation:

digital™	MICRO/PDP-11	RSX
DEC	MicroVAX	RT
DECmate	PDP	UNIBUS
DECUS	P/OS	VAX
DECwriter	Professional	VAXcluster
DIBOL	Q-Bus	VMS
LSI-11	Rainbow	VT
MASSBUS	RSTS	Work Processor

CONTENTS

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

FIGURES

CONTENTS (Cont)

TABLES

CONTENTS (Cont)

PREFACE

This reference manual provides a technically oriented description of the Network Interconnect Adapter (NIA20). This description includes detail on implementation and installation, as well as functional and logic characteristics. The NIA20 is required by Digital Equipment Corporation field service, manufacturing, engineering, software, and operational personnel.

This manual contains a preface, five chapters, and two appendices as outlined in the Table of Contents.

CHAPTER 1
INTRODUCTION

The network interconnect (NI), a major element in the network architecture of Digital Equipment Corporation, uses a single coaxial cable to transmit serial datagrams between stations (or nodes). The form of the datagrams (also called frames or packets) is dictated in the Ethernet specification.

The NI is used as a local area network (LAN) to link information processing equipment within an area (limited to a building or complex of buildings) through reliable, high-speed communications channels. Extensive and complex configurations are possible using the NI as the primary interprocessor and processor to corporate communications server products.

The NI has several advantages over other interconnects. It is a multiple-access network, not a point-to-point link. This design allows wiring and then simply tapping into the cable wherever the computing power is needed. Nodes can be added or removed from a network during its operation. There is no centralized control. Also, a system failure connected to the NI should not affect communications among other systems on the NI.

All nodes on an NI must conform to the same Ethernet rules governing data transfers and arbitration. Digital Equipment Corporation has integrated Ethernet into the Digital Network Architecture (DNA) in its DECnet products, all of which conform closely to the International Standards Organization (ISO) model for Open Systems Interconnection. Ethernet is a specification for the physical implementation of local area communications. It provides high-speed communications (10 megabits per second) and a common bus, with all nodes communicating as peers using a standard link-level protocol -- Carrier Sense Multiple Access with Collision Detection (CSMA/CD).

The CSMA/CD protocol is simple and efficient. It is a first-come/first-served system. Each node is free to communicate with any other node at any time, provided two simple rules are followed.

1. No node can start talking while another node is still talking.

2. Once the Ethernet cable is idle, if more than one node starts talking at exactly the same instant, each must stop and wait a short, randomly-determined time interval and try again.

Ethernet technology allows cable segments up to 500 meters (1650 feet) long. Through the use of repeaters, multiple segments can be connected, with up to 100 nodes each and connections separated by at least 2.5 meters (8.25 feet). LANs can be constructed with as

many as 1024 nodes, but two nodes cannot be separated by more than two repeaters.

The maximum length of coaxial cable between any two nodes is 1500 meters (4950 feet), and the maximum length of the transceiver cable (between a transceiver and the controller) is 50 meters (165 feet).

A maximum of 1000 meters (3300 feet) of point-to-point link is allowed for extending the network. One possible example would be to connect two Ethernet segments, using a high-speed, point-to-point fiberoptic link.

The 2800-meter (9240 feet) maximum end-to-end length of the LAN between any two nodes is the sum of three 500-meter (1650 foot) coaxial cable segments, plus six 50-meter (165 foot) transceiver cables, plus 1000 meters (3300 feet) of point-to-point link.

Figure 1-1 shows a large-scale configured Ethernet LAN.

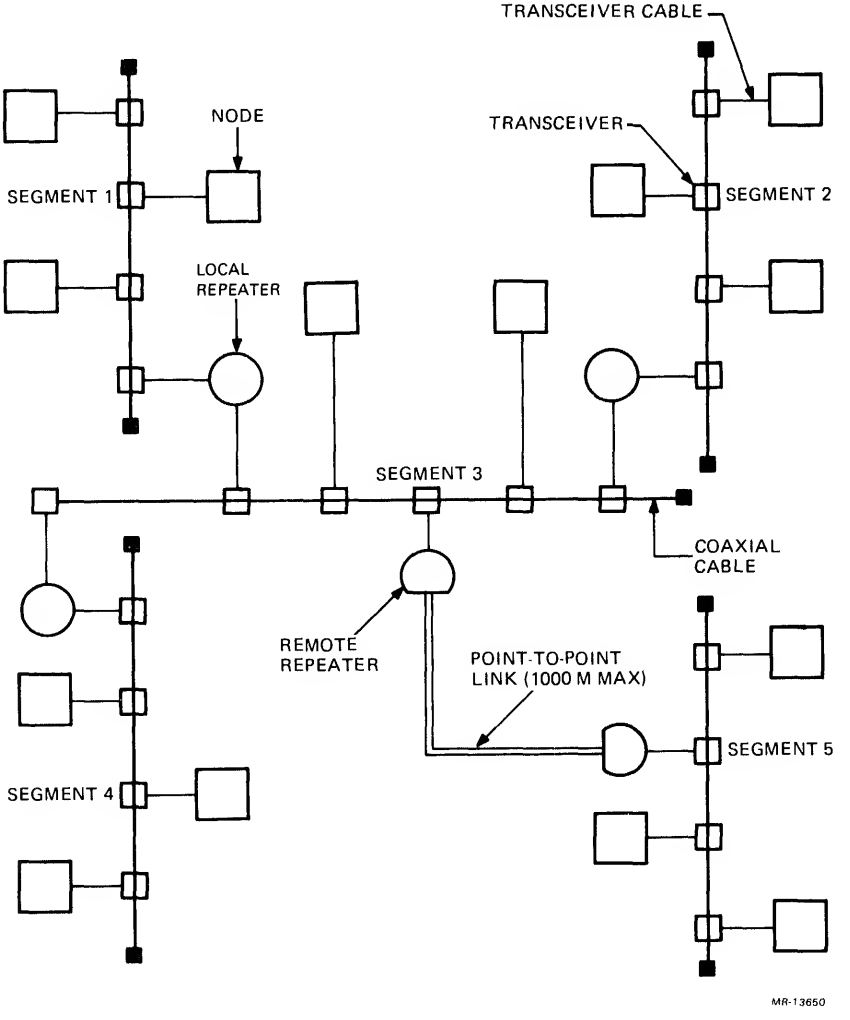


Figure 1-1 Large-Scale Ethernet Configuration

1.1 NIA20 SUBSYSTEM OVERVIEW

The Network Interconnect Adapter (NIA20) is a hardware/firmware option that will be used to interface the operating system of the KL10 to the high-speed serial NI line.

The NIA20 subsystem consists of the following:

- The port, which will reside in RH20 slot number 5. The port consists of three standard hex modules:

  M3001 -- EBus interface module
  M3002 -- Microprocessor control module
  M3003 -- CBus interface module.

  The port interfaces to the KL10 via the EBus and CBus, and to the NIA module via the port link interface (PLI) bus.

- The NIA module, which is an extended tri-board module residing in the CPU bay of the KL10. It is housed in an NI card cage and backplane unit located in the CPU bay. The NIA is controlled by the port in a master/slave manner and is used to transmit and receive data packets through the transceiver and onto the serial NI coaxial cable.

- The H4000 transceiver (and transceiver cable), which taps into the coaxial NI cable and allows the NIA to transmit, receive, and detect data collisions.

- The NI cable, which is a 10 megabit-per-second, multiaccess serial interconnect. It allows up to 1024 different nodes to communicate by exchanging data packets.

- The current limiter board (and bulkhead connector), which provides the interface between the NIA and H4000 and current-limits the 15 volts to the H4000.

The relationships among these units is shown graphically in Figure 1-2.

Although listed as part of the subsystem, the NI coaxial cable is not included as an NIA20 part because the need varies according to the configuration of the LAN.

The NIA20 is made up of the transceiver (and transceiver cable), MBus and PLI cables, current limiting bulkhead connector, card cage, and four modules: NIA module, EBus interface/port arithmetic and logic unit (ALU) module, port microprocessor control module, and CBus-PLI interface.

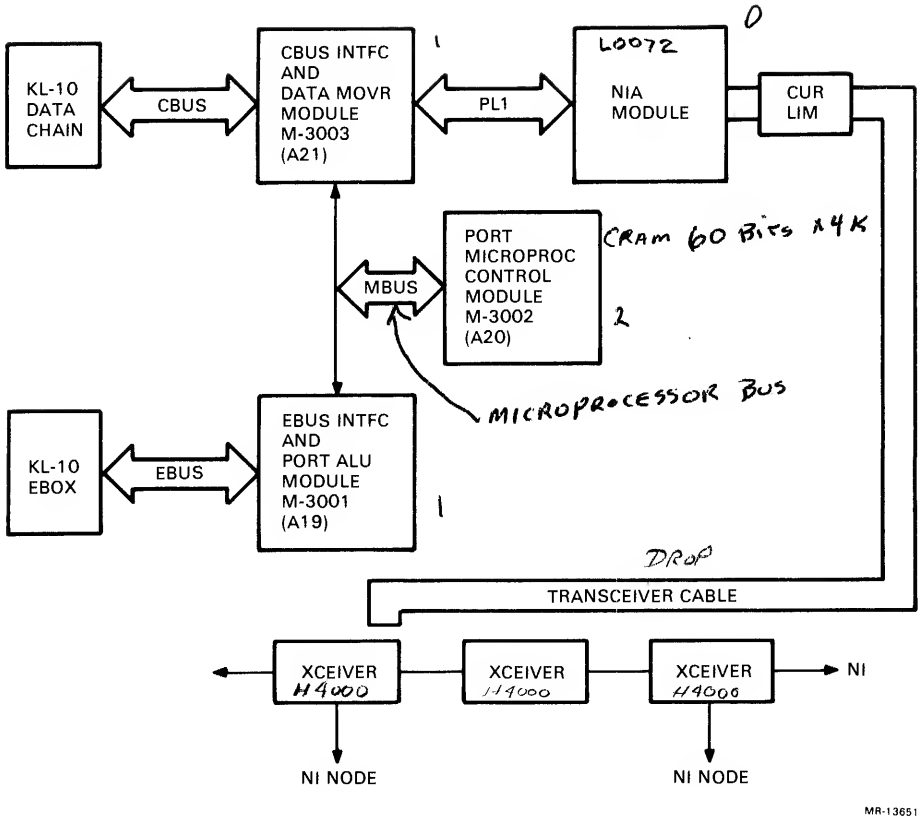


Figure 1-2 Simplified NIA20 Block Diagram

Three of these four modules (EBus interface/port ALU, port microprocessor control, and CBus-PLI interface) make up the port. The port is responsible for implementation of the data link and port layers of the Systems Communications Architecture (SCA) protocol. The port is responsible for all data transfers, between KL10 memory and the NIA module. The port gets its instructions from queued I/O structures in KL10 memory, and makes the transfers through the use of command/response queues.

1.1.1 Network Interconnect Adapter Module

The NIA module acts as the link and buffer between the KL10 PLI bus and the NI transceiver cable (between port and NI Wire). The NIA module contains a transmit buffer (2K X 10 bits), a receive buffer (16K X 10 bits) and a permanently stored ROM containing the Ethernet port address.

As the NIA module receives a packet/frame from the NI wire, the receive buffers are loaded and organized via the use of free and used buffer lists. The buffers are read by the port modules and data is sent to KL10 memory via the CBus.

The transmit buffer is loaded by the port with frame data. When the entire frame has been loaded, the port commands the NIA to transmit. When the NIA is free to transmit, it reads the transmit buffer as it transmits the frame. In addition to transmission and reception, the NIA performs the following functions:

1. CRC generation and checking
2. Transmission deferral when wire is busy
3. Detection of collisions during transmission
4. Automatic retry after collisions
5. Maintenance of transmit status conditions
6. Parallel/serial bit stream conversion
7. Encoding/decoding of the bit stream
8. Carrier sensing and data synchronization
9. Destination address decoding
10. Maintenance of receive status conditions
11. Buffering of incoming frames from 10 to 64 (depending on their size)
12. Parity generation and checking
13. Various internal loopback and checkout features.

1.1.2 Port

The port is an Am2901 based microprocessor which controls the operation of the NIA20. The port has 1K by 36 bits of local RAM memory in addition to 4K by 60 bits of control RAM. Microprogram sequencing is done by an Am2910.

Control and status information is passed between the port and the KL10 via the KL10 EBus interface. DMA data transfers between the port and the KL10 are done via the KL10 CBus interface.

The port consists of three standard hex fine-line etch modules inserted into a dedicated low-priority RH20 slot (slot 5, with slot 4 left empty) in the KL10 RH20-DTE20 backplane.

The three modules are linked by a 36-bit tristate data path called the microprocessor bus (MBus). All data is passed among the port modules via this bus.

1.1.2.1 EBus Interface/Port ALU Module (M3001) -- The EBus interface/port ALU module acts as a low-speed asynchronous control interface between the KL10 EBOX and the port. It performs all of the functions required for passing data between the EBus and the microprocessor. It also contains a 36-bit control and status register, which enables the port to control and monitor EBus operations. Most of the port protocol is processed over the EBus through this interface.

In addition, this module houses the port microprocessor ALU. The ALU consists of nine Am2901 bit slice ICs (36-bit data path) and four Am2902 high-speed look-ahead carry generators. A constant multiplexer is also included, which allows the port microprocessor to pass preassigned constants from the control store RAM (CRAM) to the ALU.

1.1.2.2 **Port Microprocessor Control Module (M3002)** -- The port microprocessor consists of a bit slice microprocessor controller, which controls the CBus/mover and the EBus/port ALU interfaces. It performs such functions as data mapping, NI packet interpretation, and some packet buffer manipulations. It contains a 2910 type microsequencer, a CRAM 4K deep by 60 bits wide, a scratch pad RAM 1K deep by 36 bits wide, a CRAM control register, and other associated control logic.

Once the CRAM is loaded and the microprocessor is started, the port is entirely under control of the microwords that are strobed from the CRAM into the CRAM control register at the beginning of each clock cycle.

1.1.2.3 **CBus-PLI Interface Module (M3003)** -- The CBus-PLI interface module acts as a high-speed synchronous DMA data transfer path and data formatter between the packet buffer and the KL10 CBus channel. This module employs a 4-bit parallel by 12-bit serial shift register for the data formatter, which is used for mapping between 8-bit bytes and 36-bit words. This module also contains the necessary control logic for performing data transfers among the shift register, the CBus, and the PLI interfaces.

In addition, the module supports a 36-bit read/write data path (MBus) between the port microprocessor and the data formatter, and an 8-bit read/write data path (PLI bus) between the port microprocessor and the PLI interface. Thus the port microprocessor transfers data direct to and from the CBus or the PLI interfaces.

The port is controlled by a four-phase clock generator located on the CBus/data mover module. One microcycle normally requires 270 ns (see Chapter 5, section 5.2, for details).

1.1.3 H4000 Transceiver

This section gives a brief overview of the H4000 Transceiver. For a full description on interfacing and operational theory, see the H4000 technical manual (Digital P.N. EK-H4000-TM-001).

The basic functions of the H4000 are as follows:

- Transmit data from the controller (NIA) onto the coaxial cable
- Receive data (including data it is transmitting) from the cable
- Detect collision conditions and notify the controller.

The transceiver also has circuitry to prevent internal problems from affecting the Ethernet (no faulty node can be allowed to affect the network adversely).

1.1.3.1 **Transmit Function** -- The transmitter amplifies data on the transceiver cable pair and transmits it onto the coaxial cable. A squelch circuit ensures that transitions on this pair are valid Ethernet 10 megabit-per-second Manchester encoded signals (not random noise) before transmitting onto the coaxial cable.

1.1.3.2 **Watchdog Timer Function** -- When the squelch circuit in the transmitter opens (permitting data to be transmitted), a watchdog timer circuit is enabled. This timer ensures that, in the event of a controller or transceiver runaway, the transmitter will be disabled within approximately 50 ms. This protects the Ethernet from being "hung." Digital's H4000 watchdog timer is triply redundant, for enhanced protection and reliability.

1.1.3.3 **Collision Presence Function** -- A collision sense circuit determines if more than one transceiver is simultaneously transmitting on the coaxial cable. A low-pass filter, which extracts the dc component of the signal on the coaxial cable, is used to make the collision determination.

1.1.3.4 **Collision Presence Test Function (Heartbeat)** -- At the end of each normal transmission, the collision presence oscillator is turned on for a short duration (pause about 360 ns and on for about 400 ns) to test the collision presence circuit. Its absence should warn the controller (NIA) that the collision-sensing circuitry in the H4000 may be faulty.

1.1.3.5 **Receive Function** -- The receiver amplifies the data received from the coaxial cable and retransmits it onto the receive pair of the transceiver cable. Squelch is provided to ensure that the receiver is enabled only when valid Ethernet signals are present on the coaxial cable.

1.1.3.6 **DC to DC Converter** -- The H4000 has a dc-dc converter to convert the +12 V to the -5.2 V and -10.2 V required by the transceiver. The converter also provides electrical isolation between the controller and the transceiver circuits and the coaxial cable, further protecting the Ethernet from node faults.

1.1.3.7 **Coaxial Cable Connection** -- The H4000 electronics module is custom fit into the casing, which also acts as the tap into the coaxial cable. The tap requires that the coaxial cable be preconditioned by drilling two small holes (opposite each other) into the outer jacket and braid.

An installation kit (H4090-KA) is available, which contains all the necessary parts, tools, and instructions needed to install the H4000.

The tap is designed to be reusable up to five times, providing that certain conditions are met (see Chapter 3, section 3.4, Installation Procedure, and Appendix A and Appendix B).

1.1.3.8 **Transceiver Cable Connections** -- The transceiver is equipped with a 15-pin D connector, meeting the Ethernet specification. The H4000 transceiver pin assignments are given in Table 1-1.

Table 1-1 H4000 Pin Assignments

Pin	Signal Name
1	Transceiver cable shield
2	Collision presence +
3	Transmit +
4	(reserved)
5	Receive +
6	Power Return
7	(reserved)
8	(reserved)
9	Collision presence -
10	Transmit -
11	(reserved)
12	Receive -
13	Power
14	(reserved)
15	(reserved)

1.2 ETHERNET SPECIFICATION OVERVIEW

This section gives a brief overview of the Ethernet specification. A full description of the data link protocol and physical details of the communications medium can be found in the Ethernet specification manual. This condensed description is provided here to show how the data is packaged and handled.

Data on the serial NI cable must conform to the Ethernet specification. The arbitration protocol used by the NI is Carrier Sense Multiple Access with Collision Detect (CSMA/CD). The service provided is a datagram class in which the data packets are delivered on a best-effort basis; no confirmation of delivery is made, and no guarantee of sequentiality or non-duplication is made. Higher levels of network software must provide these services.

1.2.1 Packet Format

Packet format is illustrated in Figure 1-3. A packet (or frame) contains the following:

1. Preamble - 64 bits (8 bytes)
2. Destination address - 48 bits (6 bytes)

3. Source address - 48 bits (6 bytes)
4. Type field - 16 bits (2 bytes)
5. Data field - (46 bytes minimum, 1500 bytes maximum)
6. Cyclic redundancy check (CRC) field - 32 bits (4 bytes).

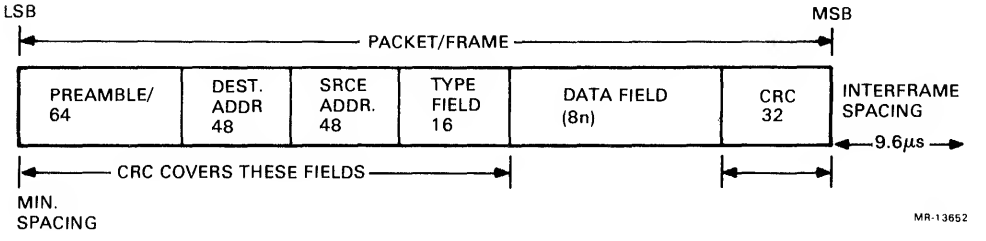


Figure 1-3 Ethernet Data Packet Format

Nodes (or stations) must be able to receive and transmit packets on the coaxial cable according to the format and spacing as shown in Figure 1-3. Each packet is a sequence of 8-bit bytes. The least significant bit of each byte (starting with the preamble) is transmitted first. The minimum packet spacing is 9.6 μs between the end of one packet and the start of another.

1.2.1.1 **Maximum Packet Size** -- 1526 bytes (8-byte preamble + 14-byte header + 1500 data bytes + 4-byte CRC). The header is made up of the destination and source addresses and the type field.

1.2.1.2 **Minimum Packet Size** -- 72 bytes (8-byte preamble + 14-byte header + 46 data bytes + 4-byte CRC). Any received bit sequence smaller than the minimum valid packet (with minimum data field) is considered to be a collision fragment and is discarded. These small collision fragments are also called runts.

1.2.1.3 **Preamble** -- This is a 64-bit synchronization pattern (alternating 1s and 0s and ending in two consecutive 1s). Synchronization and stabilization occur during the preamble, and two 1s at the end indicate the start of coded data. If two consecutive 0s are detected, an error must have occurred, and the receive link management component of the data link blocks all further bits of the current frame.

1.2.1.4 **Destination Address** -- A 48-bit field that specifies the station(s) to which the packet is being transmitted. Each station examines this field to determine if it should accept the packet. The first bit (1 sb) transmitted indicates the type of address:

If the least significant bit (LSB) is a 0, the field contains the unique address of the one destination station.

If the LSB is a 1, the field specifies a logical group of recipients.

A special case is the broadcast (all stations) address, which is all 1s.

1.2.1.5 **Source Address** -- This 48-bit field contains the unique address of the station that is transmitting the packet.

1.2.1.6 **Type Field** -- This 16-bit field is used to identify the type of higher level protocol associated with the packet. This field determines how the data field is interpreted.

1.2.1.7 **Data Field** -- This field contains an integral number of bytes ranging from 46 to 1500. (The minimum ensures that valid packets will be distinguishable from collision fragments.)

1.2.1.8 **Packet Check Sequence** -- This 32-bit field contains a cyclic redundancy check (CRC) code. The CRC checks the address (destination and source), type, and data fields. A simplified explanation is to have both the transmitter and receiver (using the same algorithm) calculate a 32-bit polynomial, based on the covered data. The transmitter sends its CRC. The receiver composes the CRC it calculated with the CRC it received. The two CRCs must match; if not, an error has occurred.

1.2.1.9 **Round-Trip Delay** -- The maximum 2800 meter end-to-end, round trip delay for a bit is 51.2 μs.

1.2.2 Control Procedures

The control procedures determine how and when a host station may transmit packets to the common cable. The main purpose is to establish fair resolution of occasional contention among transmitting stations. If multiple stations attempt to transmit at the same time (transmission overlap), the result is a collision.

NOTE

Only transmitting stations can recognize a collision. Normal transmissions have an ac component varying on a dc level, but when two or more stations transmit at the same time, the ac component varies about a dc level that is approximately twice the normal dc level. This activates the collision detect signal.

1.2.2.1 **Defer** -- A station must not transmit to the coaxial cable when a carrier (message from another station) is present or within the minimum packet spacing time after a carrier has ended (9.6 μs).

1.2.2.2 Transmit -- A station may transmit if it is not deferring. It may continue to transmit until the end of the frame is reached or a collision is detected.

1.2.2.3 Abort -- If a collision is detected, transmission of the packet must stop and a jam (4-6 bytes of arbitrary data) is transmitted to ensure that all other participants in the collision also recognize its occurrence.

1.2.2.4 Retransmit -- After a station has detected a collision and has aborted, it must wait for a random retransmission delay, defer as usual, and then attempt to retransmit the packet. The random time interval is computed using the backoff algorithm given in Section 1.2.2.5. After 16 transmission attempts, a higher level (such as software) decision is made to determine whether to continue or abandon the effort.

1.2.2.5 Backoff -- Retransmission delays are computed using the truncated binary exponential backoff algorithm, with the aim of resolving contention fairly among as many as 1024 stations. Each station has a 10-bit random number generator. After a collision, each station looks at the first bit of its generator (a one-bit window). If the bit is a 1, transmit. If the bit is a 0, wait one slot time (51.2 μsec) and transmit. This gives two contending stations a 50 percent chance of success. If the second attempt fails, both stations look at the first two bits of their generators (a 2-bit window) and wait the indicated number of slot times (0,1,2, or 3), thereby reducing the chance of collision to 25 percent. This procedure continues until each station is looking at all 10 bits. Since the maximum random number is 1023 (10 bits), retransmission attempts 10 through 15 use the entire 10-bit number, representing some random number between 0 and 1023.

1.2.3 Manchester Encoding

Manchester encoding is used on the coaxial cable. It has a 50 percent duty cycle and ensures a transition in the middle of every bit cell (data transition). The first half of the bit cell contains the complement of the bit value, and the second half contains the true value of the bit. The data rate is 10 megabits per second = 100 ns per bit cell (see Figure 1-4).

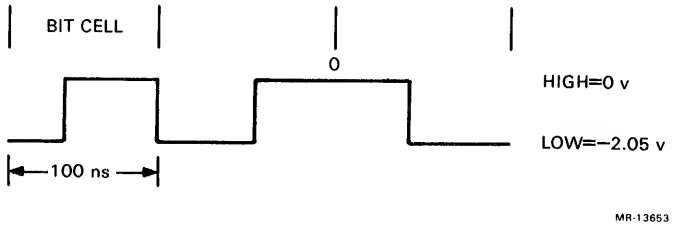


Figure 1-4 Manchester Encoding

1.2.4 Carrier Sense

The presence of data transitions indicates that carrier sense is present. If a transition is not seen between 0.75 and 1.25 bit times since the center of the last bit cell, then carrier sense has been lost (indicating the end of a packet).

Carrier sense is derived from any data transitions detected on either the receive detect or collision detect circuitry within the last 160 ns.

1.2.5 Transceiver Connections

Up to 100 transceivers may be connected to the coaxial cable. Each transceiver has a built-in tap, which can be attached or removed from a network while it remains in operation. The taps must be attached only at the special points marked by the cable manufacturer, at 2.5 meter intervals.

1.2.6 Transceivers

Individual transceiver devices depend on their manufacturers. They have taps that provide the interface into the coaxial cable, and, as a general rule, they have an interface to the host system via shielded cable and a 15-pin connector. Transceivers provide the host with the following signals (minimum standard type unit):

Shield connection
Power pair connection
Transmit pair connection
Receive pair connection
Collision pair connection
Remaining pins are reserved.

CHAPTER 2
IMPLEMENTATION

This chapter deals with various elements involved in the implementation of the NI on large computer group (LCG) systems. The NIA20 uses a queued I/O structure with the NIA20 handling the actual I/O between the NI and KL10 memory. This structure involves concepts and protocols new to LCG systems.

2.1 IMPLEMENTATION OVERVIEW

There are two mechanisms used in communication between the KL10 (port driver) and the NIA20 port (port microprocessor). In the first, the port command and status register (CSR) can be read and written by the KL10 over the EBus. The port can also use the EBus to read and write KL10 memory. In the second method, the queued protocol is used for both port control and data transfer. The CBus is the data path to and from KL10 memory when the queued protocol is used. The EBus is used for control and status information and interrupts when using queued protocol.

In the queued protocol, the NIA20 port takes its instructions from a doubly linked command queue in KL10 memory. Responses are put to a response queue. The KL10 links commands to the tail of the command queue, and the port removes commands from the head of the queue and processes them. The port links command responses to the response queue at the tail, and the KL10 removes entries from the response queue at its head.

Other queues used by the port and the KL10 implement the NI data transfer mechanisms. The use of these queues depends on the type of protocol used to assemble the data. There are up to 16 protocol type free queues (if all are enabled) and one unknown protocol type free queue. The locations of the protocol-dependent free queues are discussed in sections 2.2 and 2.3.1.

The free queues are used as repositories of queue entries (see Figure 2-1). When either the port or the KL10 needs an entry to link to a queue, it gets the entry from the appropriate free queue. When a queue entry has been processed by either the port or the KL10, the entry is put back into the free queue. Thus, the total number of queue entries remains fixed.

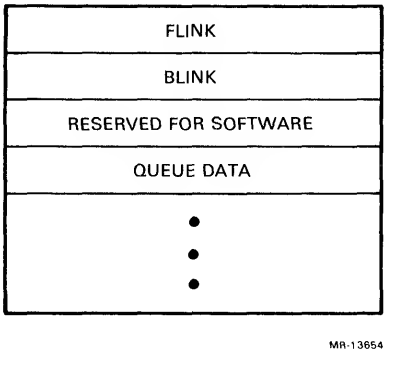


Figure 2-1 Queue Entry Format

Sometimes, the term process is used when explaining operations on queues or commands. This convention will apply when the operations are the same, whether the CPU or the port performs them.

2.2 QUEUE STRUCTURE

The queues used by the NIA20 are doubly linked lists of commands and responses. The queues reside in KL10 physical memory and are used by both the port driver software of the KL10 and the port microcode.

Text data out, from the port driver to the port, appears after the command portion of command queue entries. Text data in, from the port to the port driver, appears after the response portion of response queue entries. Status information may or may not have appended data.

2.2.1 Queue Linkage

Each queue entry contains two pointers (see Figure 2-2). One, the forward link (flink) points to the next queue entry. The other, the backward link (blink) points to the previous queue entry (to its flink). Both pointers actually point to the flink of the next entry, never to a blink.

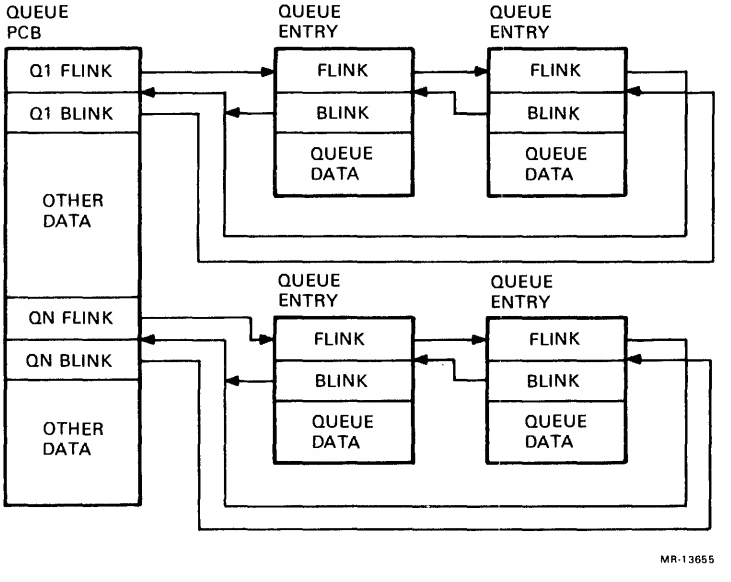


Figure 2-2 Queue Linkage

Any entry on the queue can be accessed from any other entry by using one or both of these pointers. The pointers are followed from entry to entry until the desired queue entry is found. This can be done in either direction from the original entry by using either the flinks or the blinks to go through the queue.

The queues all originate in the port control block (PCB). Both the KL10 and the port gain access to queues through the PCB. The PCB, located in KL10 physical memory, is used to control access to the queues and to set up certain queue parameters.

The PCB entries for each queue do not have the same format as a queue entry. There is no information field in the PCB entries for a queue. The PCB entries are queue headers (see section 2.3.1).

2.2.2 Queue Interlocks

To prevent the KL10 and the NIA20 from accessing a queue at the same time, there is an interlock word in the PCB associated with each queue. To link an entry to a queue or to remove a queue entry, the device performing the operation must first obtain the interlock word.

If the queue is available, the interlock word is set to a value of -1 (all 1s). When attempting to gain access to a queue, the interlock word must be incremented and tested for a value of 0. If the value after incrementing is 0, the interlock has been obtained and the requesting process can manipulate the queue. If the word is positive, the queue is interlocked by another process and access is prohibited.

The interlock is only a "good will" access control mechanism. There is no hardware interlock to prevent queue access by processes that ignore the condition of the interlock. If two processes manipulate the queue at the same time, linkage could be lost, and the results would be unpredictable.

The queue should be interlocked only for the time required to insert or remove an entry. When the process manipulating the queue is complete, with the insertion or removal, it must leave the interlock word with a value of -1.

2.2.3 Queue Locations

The PCB is used to anchor the queues at a known point in the host memory and to provide certain initial parameters to the port. The NIA20 (port) must be told, by the port driver, where the base address of the PCB is located.

Before the NIA20 is initialized, the port driver must set up the channel to transfer the base address of the PCB to the port by setting up a channel command word (CCW) to transfer three words starting with word 24 octal of the PCB.

The port driver must also build, in word 24 octal of the EPT, a jump word to word 27 octal of the PCB. The contents of the CCW in the PCB must specify a three-word forward data transfer and halt command with a starting address of word 24 octal of the PCB.

At initialization, the port will start the channel with a CBus Start, which reads the contents of these locations. This provides the port with the base address of the PCB and two words not currently defined, but reserved for future use.

Figure 2-3 shows the format of the PCB as it appears in KL10 memory.

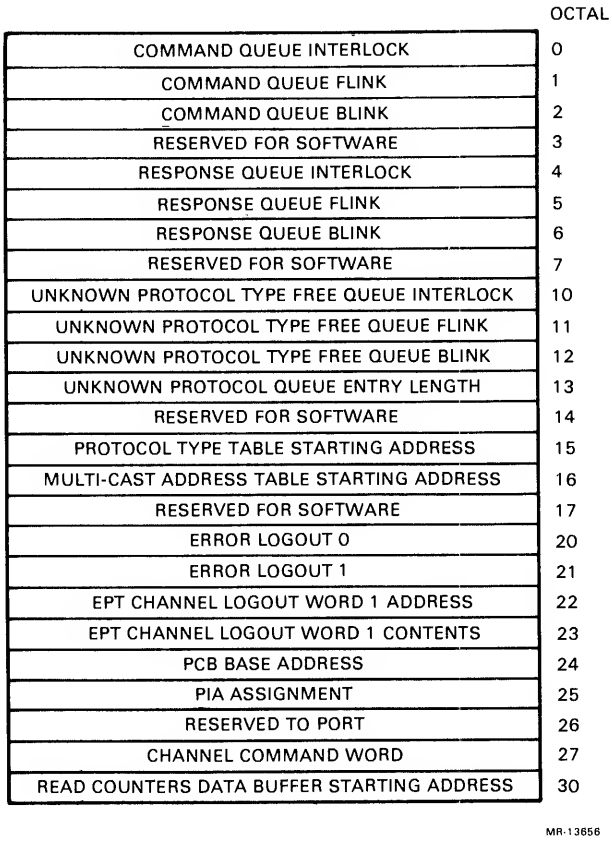


Figure 2-3 KL10 Memory PCB Format

Word 22 of the PCB is written by the port during initialization time with the address of the EPT channel logout word 1, which the port gets from the port driver software. Words 22 and 23 of the PCB are used by the port during channel error recovery.

Word 23 contains the channel logout word 1 written by the port on any kind of channel error detected during or immediately after a direct memory access (DMA) transfer. The format of error word 3 is shown in Figure 2-4, and Table 2-1 provides bit descriptions.

Word 24 octal of the PCB is the address of the first word of the PCB; the NIA20 has no other way of finding the PCB.

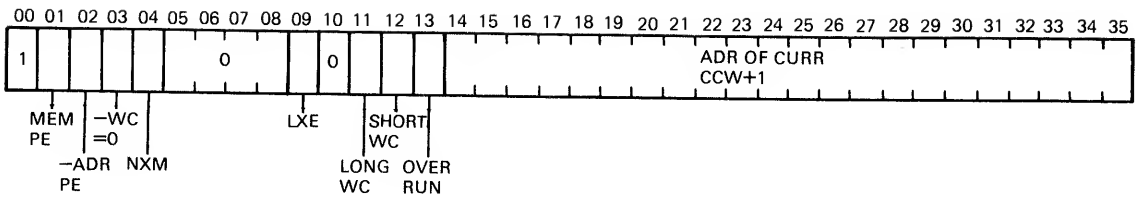


Figure 2-4 Error Word 3 Format

Table 2-1 Error Word 3 Bit Descriptions

Bits	Name	Description
01	MEM PE	Memory parity error
02	-ADR PE	Not address parity error
03	-WC=0	Channel word count did not = 0 when channel did a store to EPT
04	NXM	Channel ref nonexistent memory
09	LXE	Error detected after port term transfer. Channel aborts next transfer
11	Long WC	Port completes transfer, but word count in CCW not reached
12	Short WC	Channel transferred data specified by CCW, but port still has data
13	Overrun	If device read, port sent data but channel buffers were full
		If device write, port requested data but channel buffers were empty

Word 27 octal is reserved for the CCW, where the port writes a CCW-style word over the EBus when it wishes to transfer data over the KL10 CBus. The port driver is responsible for writing a channel jump word into the appropriate EPT location corresponding to the RH20 backplane slot (slot 5) where the NIA20 is installed.

Word 25 is always reserved for port microcode use. The port driver should never write this location or depend on its value.

Word 30 of the PCB is a pointer to the beginning of the read counter data buffer. This address is supplied by the port driver software at initialization.

When the NIA20 is being initialized, the port driver must set up the channel to transfer the contents of the PCB to the port. This is done by setting up a CCW to transfer three words starting with word 24 of the PCB, from KL10 memory to the channel. The port will start the channel and read the contents of these locations. This setup provides the port with the base of the PCB, and its physical interrupt assignment (PIA).

Since the port will be using the channel to transfer large blocks of data, the channel control logic will be writing logout information into the executive process table (EPT). An error that the channel discovers will be reported in the usual manner via the EPT. Normal operation of the port will generate many long word count errors, due to incoming frames of indeterminate size. These errors may be ignored.

Command queue and response queue locations are obtained directly from the PCB. Use of the protocol type table (PTT), (see Figure 2-5), multicast address table (MCAT) (see Figure 2-6), and unknown protocol type free queue is less direct. Received datagrams will be node address filtered prior to entry into the NIA receive buffer. The buffered datagrams will be multicast filtered, if their destination address is a multicast address. Protocol type filtering determines which queue to use for handling the frame.

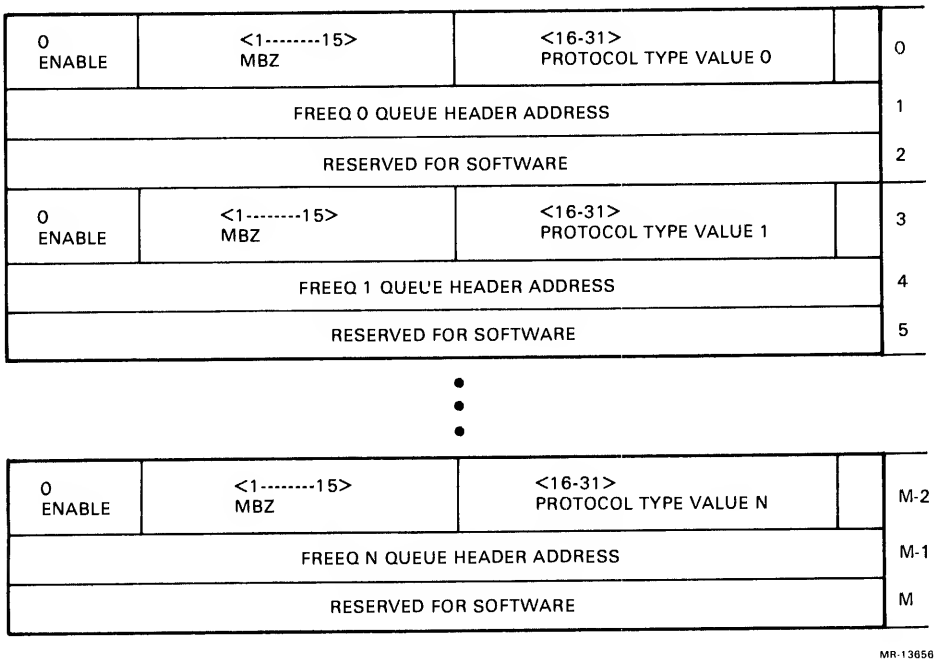


Figure 2-5 Protocol Type Table Format

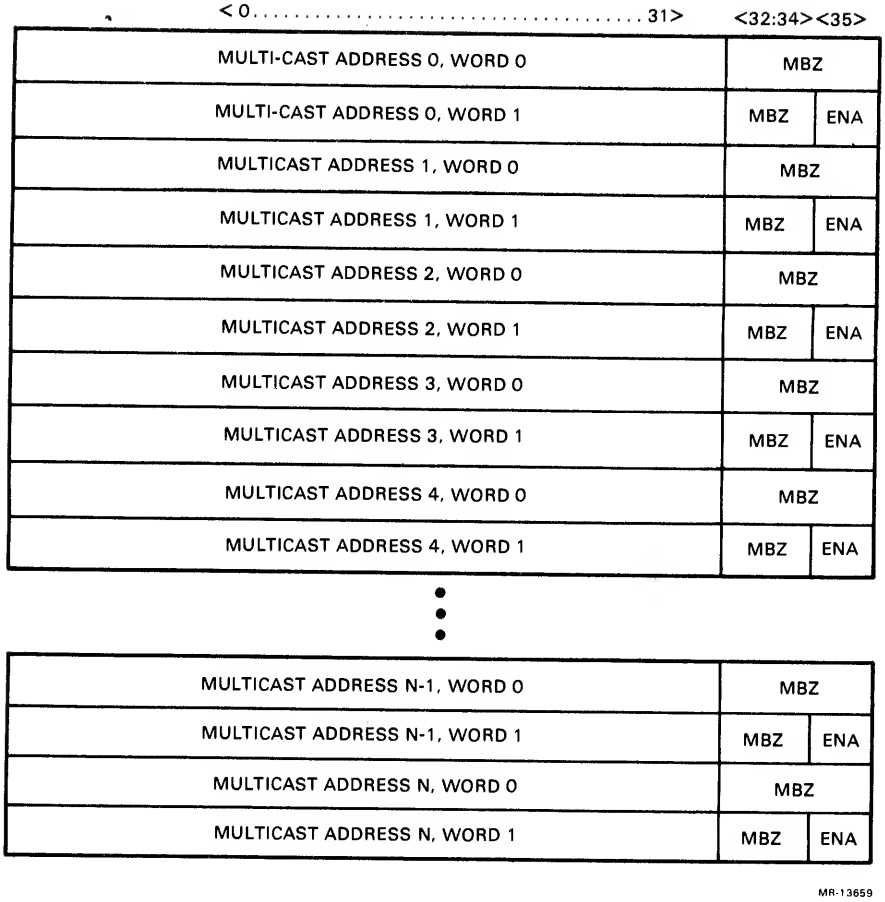


Figure 2-6 Multicast Address Table Format

All received datagrams must pass address filtering in the following manner: If the destination address is -1 (all 1s), it is a broadcast message and the port accepts it. If the destination address has bit 47 a 0, it is a specific physical address. If this specific physical address matches the port address, the port accepts it. In both cases, the accepted datagram is filtered for protocol type.

Protocol type filtering occurs as follows: When a datagram is received, the protocol type is checked by the port against the table of enabled protocol types. The PCB has the starting address for the protocol type table. If a match is found in this table, the associated pointer is used to get the queue entry to store the received datagram. If a match is not found, the required queue entry is obtained from the unknown protocol type free list queue anchored in the PCB.

If the datagram destination address has bit 47 a 1 (and not all 1s) it is a multicast address. Further address filtering occurs as follows: The address is checked by the port against the table of enabled multicast addresses. The PCB has the starting address for

the MCAT. If a match is found, protocol type filtering occurs. If no match is found, the datagram was not intended for this port and is discarded.

Both the PTT and MCAT have an enable bit. The bit must be set for the entry to be considered valid. Both tables are arranged in ascending order and both tables are allocated beginning on a four-word block. The tables are cached within the port when the appropriate command is issued.

2.3 QUEUE HANDLING

The port driver and the port modify the queue with the port driver putting commands on the command queue and the port removing and processing the commands. The port puts responses to commands on the response queue and the port driver of the KL10 removes the responses.

When the port driver needs a queue entry, it removes the entry from the appropriate free queue. It builds a command and links the command to a command queue.

The port removes commands from the head of the command queue and processes them. After processing, the queue entry containing the command is either linked to the response queue or back to the free queue.

2.3.1 Queue Headers

The PCB contains queue headers to anchor the command queue, the response queue, and the unknown protocol type free queue. The PCB also contains base pointers to the MCAT and the PTT.

Queue headers anchor a queue structure. A queue header may be located in the PCB or in the host memory as a free-standing queue structure (used by the PTT as the header for the different type queues). Queue headers are made up of a queue interlock word, queue flink, queue blink, and a queue length word as shown in Figure 2-7.

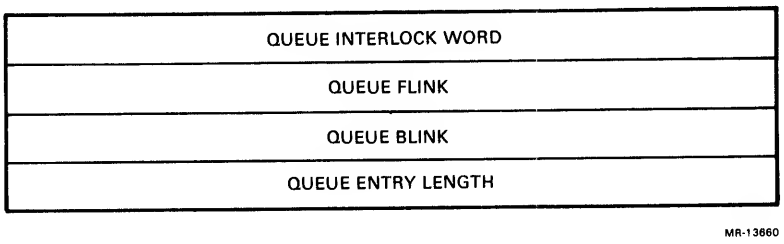


Figure 2-7 Queue Header Format

The queue entry length specifies the length, in words, of the queue entries on the queue. This includes the flink, blink, interlock and operation code words in addition to the data. All entries obtained for a queue are of the same length.

The example shown in Figure 2-8 uses both the PCB queue headers and a freestanding queue header.

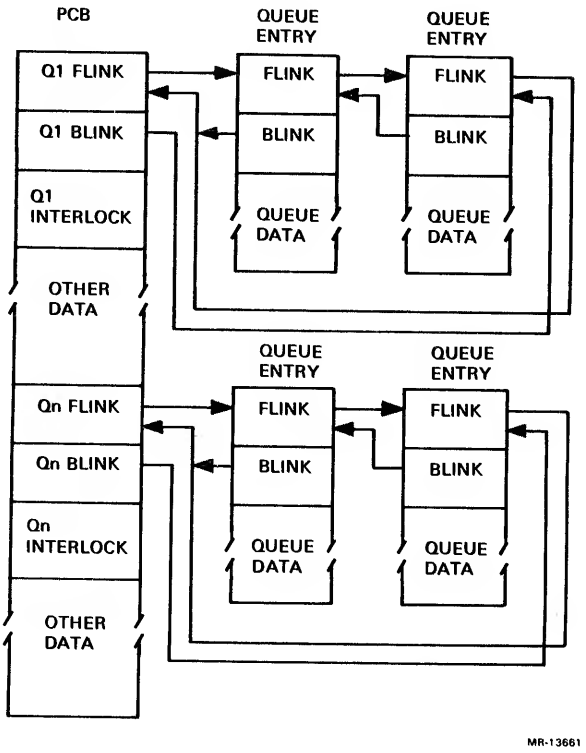


Figure 2-8 Use of Queue Headers

2.3.2 Entry Linking

To link an entry to the tail of a queue, the interlock word for the queue must be obtained. Then the process uses the blink word of the PCB to find the tail of the queue.

The flink of the entry at the tail is changed to point to the new entry. The blink of the PCB is also changed to point to the new entry. The flink of the new entry is set to point to the flink of the PCB entry for the queue. The blink of the new entry is set to point to the old tail entry.

When the new entry is linked to the queue, the process must then release the queue interlock by setting it to -1.

An empty queue specified by its header at address H is shown in Figure 2-9.

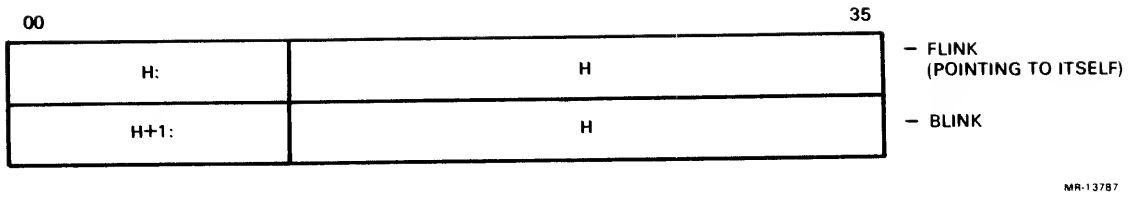


Figure 2-9 Empty Queue

If an entry at address A is inserted into an empty queue, the queue is as shown in Figure 2-10.

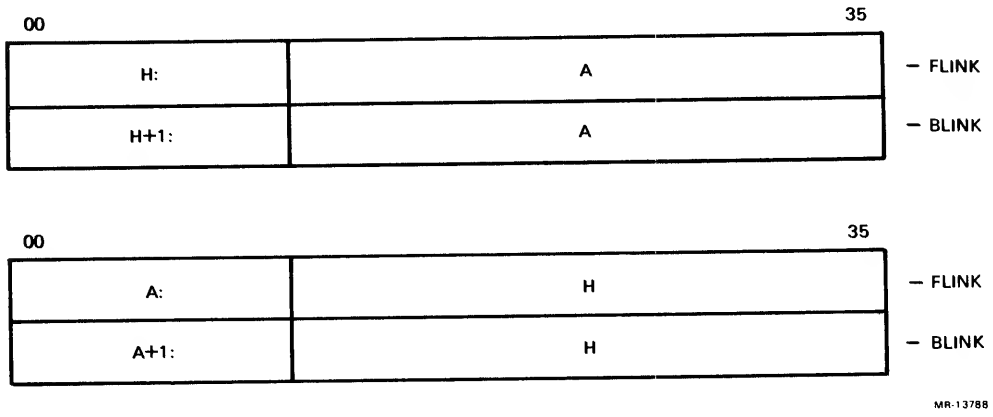


Figure 2-10 Queue with Entry at Address A

If an entry at address B is inserted at the tail of the queue, the queue is as shown in Figure 2-11.

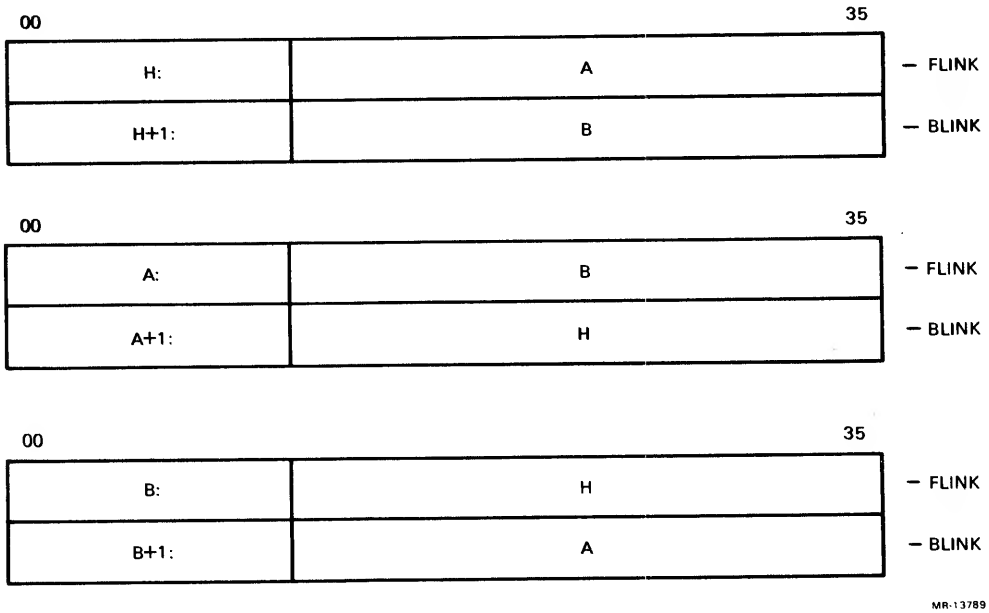


Figure 2-11 Entry at Address B at the Tail of the Queue

If an entry at address C is inserted at the tail, the queue appears as in Figure 2-12.

2.3.3 Entry Removal

To remove an entry from the head of a queue, the process must obtain the interlock word for the queue. Then the process uses the flink from the PCB to find the first queue entry.

The flink in the PCB is made equal to the flink of the entry. The flink in the PCB is now pointing to the second entry on the list. The blink in the second entry is then changed to point to the flink in the PCB.

The queue entry has now been removed and the only pointers to the entry are in the process that removed it. The process can then set the interlock word in the PCB to -1 so that another process can gain access to the queue. The process that removed the entry can manipulate the data areas of the entry.

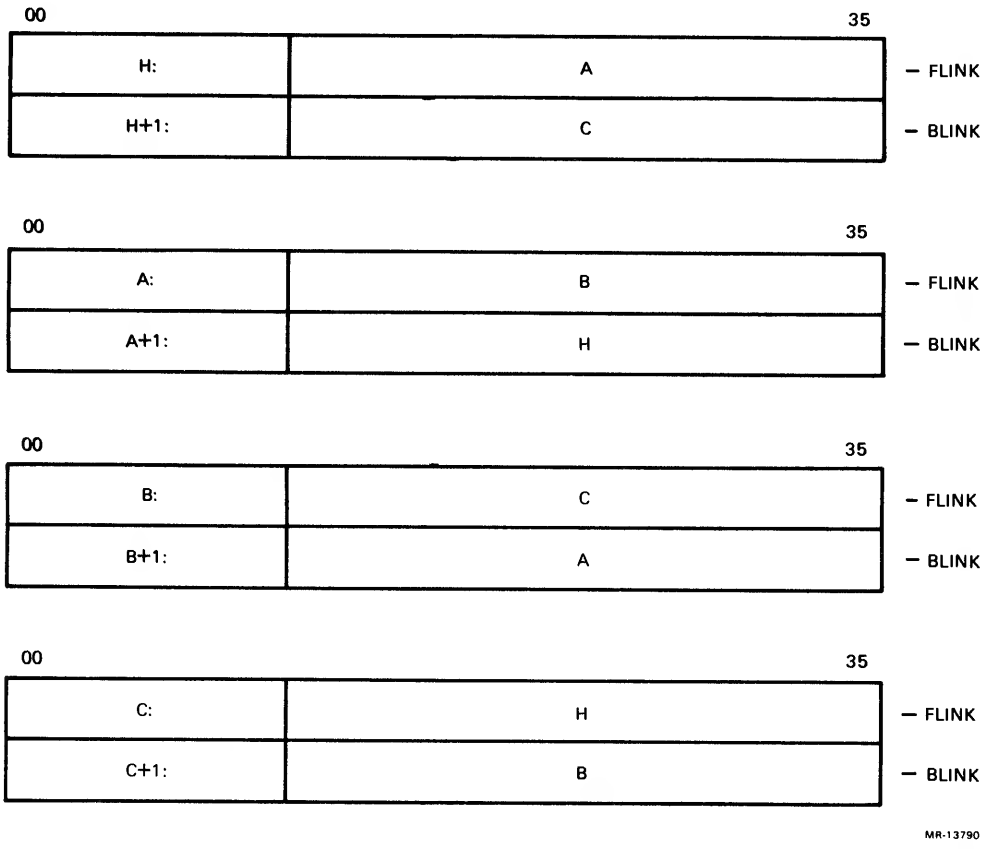


Figure 2-12 Entry at Address C at the Tail of the Queue

In the previous example, with the queue containing entries A, B, and C, the entry at A can be removed, giving the queue illustrated in Figure 2-13.

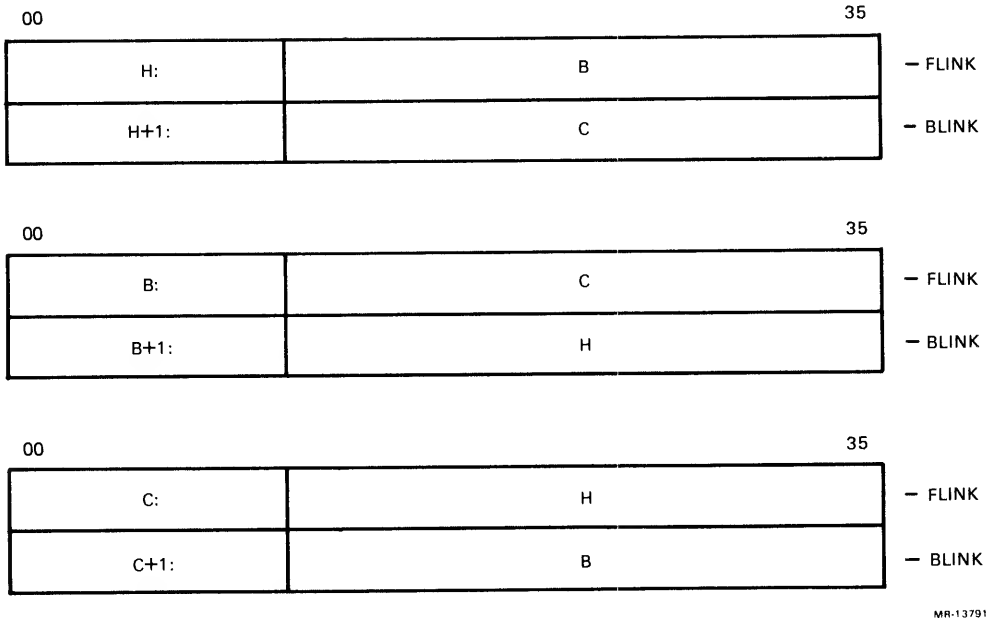


Figure 2-13 Queue Containing Entries A, B, and C, Where A Can Be Removed

2.3.3.1 Buffer Segment Descriptors -- A buffer segment descriptor (BSD) is a construct that is convenient for the network software (for example, DECnet). It is used to describe a segment of KL10 memory and gives the software the ability to append layering or handling data to a message without rearranging the entire message or creating overhead by copying the datagram from buffer to buffer.

A buffer consists of a list of physically contiguous segments of memory. Different segments of a buffer are not assumed contiguous and may be anywhere within the physical address space of the host. It is assumed that different segments of a buffer are unique (that is, they do not overlap) within a host. A buffer is described by a list of BSDs.

Each BSD describes a single contiguous piece of a buffer. Each descriptor is a four-word block, allocated on a four-word boundary, and built by the driver in physical host memory. In each of these blocks is a pointer to the next descriptor block (if any), a pointer to the segment of the buffer so described, and a field indicating what packing mode the segment is in. In addition, within each block is a field giving the size of the buffer segment, in bytes.

A buffer is referenced by giving the physical address of the first word of the first BSD in the chain of BSDs. This address is called out in the send datagram command packet, when the flags byte of that command indicates that BSDs are in use.

A flags bit in the command format for a send datagram command, when clear, indicates that the datagram to be sent is in immediate mode, meaning that the data to be transmitted follows the destination address in the command queue entry. The same flags bit, when set, indicates the use of a BSD.

BSDs are used whenever it is necessary to send different sections of a message or when different programs build different sections of the datagram -- helpful for messages passing down the layers of the network software.

2.3.3.2 Buffer Segment Descriptor Format

The BSD format is specified in Figure 2-14. The packing mode field describes the packing mode of the buffer segment pointed to. The packing modes, with the associated bit values are 0 -- Packing mode = Industry compatible, and 1 -- Packing mode = Reserved (RSVD).

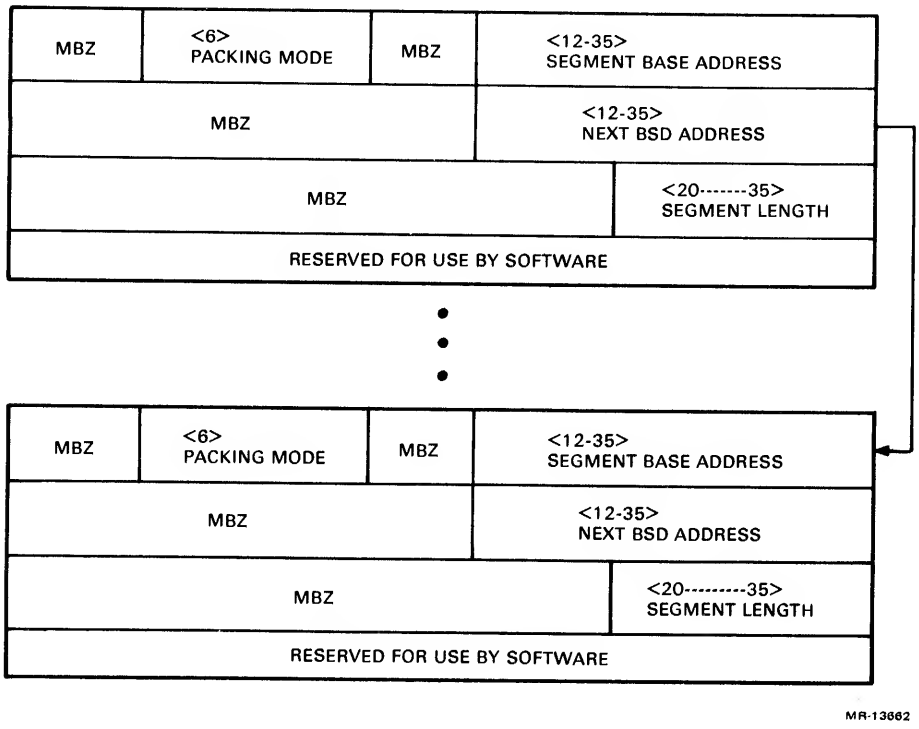


Figure 2-14 BSD Format

The segment base address field gives the physical address of the buffer segment described. The buffer segment must begin on a whole word boundary. The next BSD address points to the first word of the next buffer segment descriptor in the chain. If the next BSD field is 0, there is no next segment descriptor.

The segment length field gives the length in bytes of the buffer segment pointed to.

It is standard use for BSDs to be built in the queue entries on the command queues. BSDs must be allocated on a four-word boundary. For incoming datagrams there is only BSD-type processing.

2.4 COMMANDS AND RESPONSES

Communication between the port driver and the port is accomplished by using command packets and response packets. A command is a request from the port driver to the port. Placing a command on the command queue initiates processing in the port.

A response is a packet from the port to the driver, informing it of an event. The event may be one of the following:

An error occurred while processing a command. This always causes a response.

A response to a command which had the response bit set in the flags field. The port must send the driver a response if the original command had the response bit set.

The reception of an incoming packet from the wire.

If the response queue is empty when an entry is added by the port, a non-vectored interrupt is sent to the host (to indicate that the driver look at the response queue). If the response queue was not empty, the entry will be seen by the driver as it sequentially processes the queue.

After processing a command, the port always checks to see if another exists on the command queue. If not, the port goes idle. The port wakes up again when the port driver places a command in the command queue, and writes the command queue available bit in the control status register (CSR) with a 1 to indicate that a new command is available.

The commands available to the port driver are listed below. The port can build eight possible responses to the commands and can generate one other response -- for datagrams recieved from Ethernet.

1. Send Datagram (SNDDG) causes an NI datagram to be built and transmitted as an Ethernet packet. The command may or may not use buffer segment descriptors. The port response

is datagram sent (DGSNT). The format of the response, if requested, follows the format of the command (i.e., BSD or non-BSD).

2. Datagram Received (DGRCV) can be only a response, notifying the port driver that the port has received a packet over Ethernet. This response is always in BSD format.

3. Load Protocol Type Table (LDPTT) causes the PTT to be internally cached by the port. The port response is protocol type table loaded (PTTLD).

4. Load Multicast Address Table (LDMCAT) causes the MCAT to be cached internally by the port. The port response is multicast address table loaded (MCATLD).

5. Read and/or Clear Counters (RCCNT) causes all the event counters kept by the port microcode to be read if the response bit in the flags field is set. If bit 14 in the flags field is set, all the counters are cleared. This command is a performance monitoring and diagnostic feature. The port response is counters read and/or cleared (CNTRC).

6. Write PLI (WRTPLI) causes a PLI write function with the specified control bits and data byte. This command is a diagnostic feature. The port response is port link interface written (PLIWRT).

7. Read PLI (RDPLI) causes a PLI read function with the specified control bits. If a response is built for this command, the data byte read from the PLI is returned. This command is a diagnostic feature. The port response is PLI read (PLIRD).

8. Read NI Station Address (RDNSA) reads the NI station address from the NI link physical address ROMs. This command also reports the state of several link mode bits. The port response is NI station address read (NSARD).

9. Write NI Station Address (WRTNSA) writes the NI station into the NI link address RAMs. Also sets the state of several link mode bits. The port response is NI station address written (NSAWRT).

2.4.1 Command and Response Formats

The following sections describe the format of the various commands and responses that the port receives and produces.

2.4.1.1 Send Datagram (SNDDG) Command -- To send a datagram, the KL10 gets an entry from the associated free queue and builds the datagram by putting the data into the queue entry. The queue entry

is then put onto the command queue as a send datagram (SNDDG) command. The port de-links commands from the head of the command queue and processes them. When the send datagram command reaches the head of the queue it is processed by the port.

At the completion of the command SNDDG, the port microcode will determine if it should build a response queue entry for this command. If the response bit is set in the flags words, the port will build a response entry. If the response bit is off for this particular command, no response will be built unless an error occurred during the processing or transmission of the packet. Any error condition always causes the port to build a response packet.

The format of the response packet is similar to the send datagram packet; the difference is that the status field is returned with a non-0 value to indicate the conditions or type of failure. Refer to Section 2.4.1.2 for a detailed description of all of the allowable values for the status field. The response is called a datagram sent response (DGSNT).

If no response is to be built, the queue entry containing the SNDDG command will be linked to the tail of the associated free queue by the port. The queue entry is then available for reuse.

A particular format applies if the flags byte specifies BSD usage. If the BSD format is used, data bytes are transmitted onto the NI wire from the BSD left to right. That is, for a given word, the data in bits 0-7 is transmitted first, the data in bits 8-15 is transmitted second, the data in bits 16-24 is transmitted third, and so on. This left-to-right format is the norm for all data transmitted from a BSD.

NOTE

Buffers and text length fields must always describe full bytes. It is illegal for a buffer to terminate in the middle of a byte. The result, if this restriction is violated, is undefined. It is legal for a BSD to terminate in a half-byte, under certain conditions.

The format of this command as a command queue entry is specified in Figures 2-15 and 2-16.

The command status fields are zero. The response to a command will have a non-zero status field (see DGSNT response for bit definitions).

The operation code for a send datagram command is a 1.

The format of the flags field for all commands is shown in Figures 2-17 and 2-18. There are two formats; one for the SNDDG command and response (and the DGRCV response), and one for all other commands and responses.

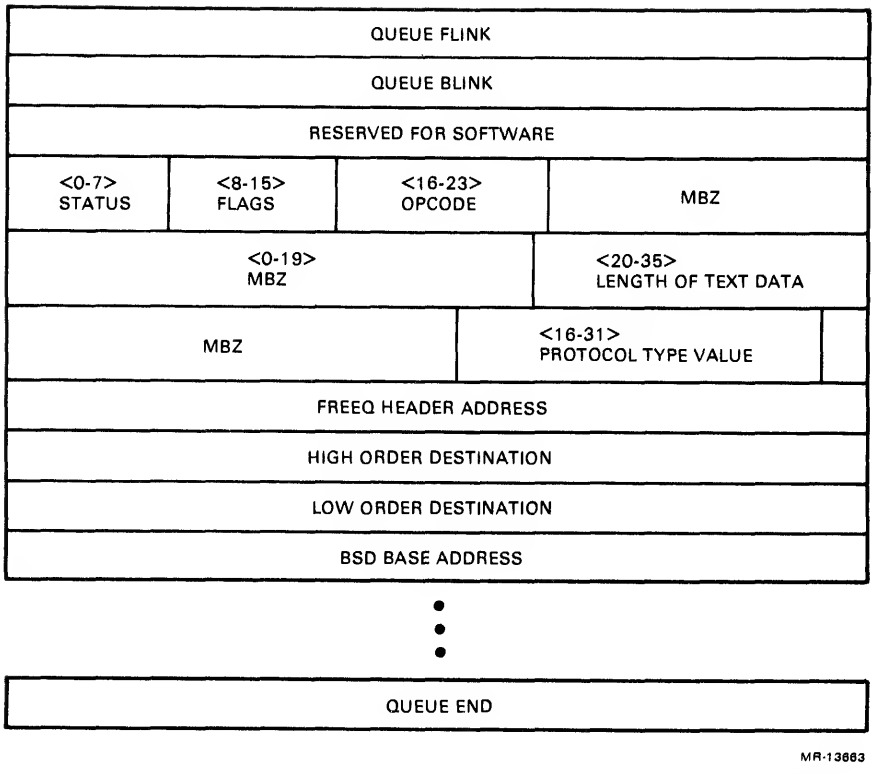


Figure 2-15 SNDDG Command Format (BSD)

The 0-pack (packing format) field defines the packing format for non-BSD datagrams: 0 = Industry compatible, 1 = Reserved.

When the ICRC is set, the port driver has appended a four-character CRC at the end of the datagram. This CRC is to be used instead of the internally generated CRC. This feature must be used in the transmission of self-directed datagrams, because the design of the NI adapter shares the CRC generator/checker between receive and transmit circuits. The length fields in the command packets do not reflect the addition of the four extra characters.

When the pad field is set, the port will pad packets to be sent that are less than the Ethernet minimum size by appending the remaining bytes with zeros to the minimum packet size. In addition, two bytes indicating the valid data length will be prepended to the text data. These length bytes are transmitted low-order byte first, and indicate the number of actual text bytes (not including padding) that occur within the packet. The two-length bytes are always transmitted when padding is enabled, whether or not padding was necessary for a particular packet transmitted. When clear, the port does not do such padding. If a packet is padded, it will be padded with zeros. If a packet is presented to the port without this bit set, and it is less than 46 bytes, an error response for a runt (less than the minimum legal size of 46 bytes) packet is generated. Packets presented that are larger than the maximum Ethernet packet size always cause a length error and generate an error response packet.

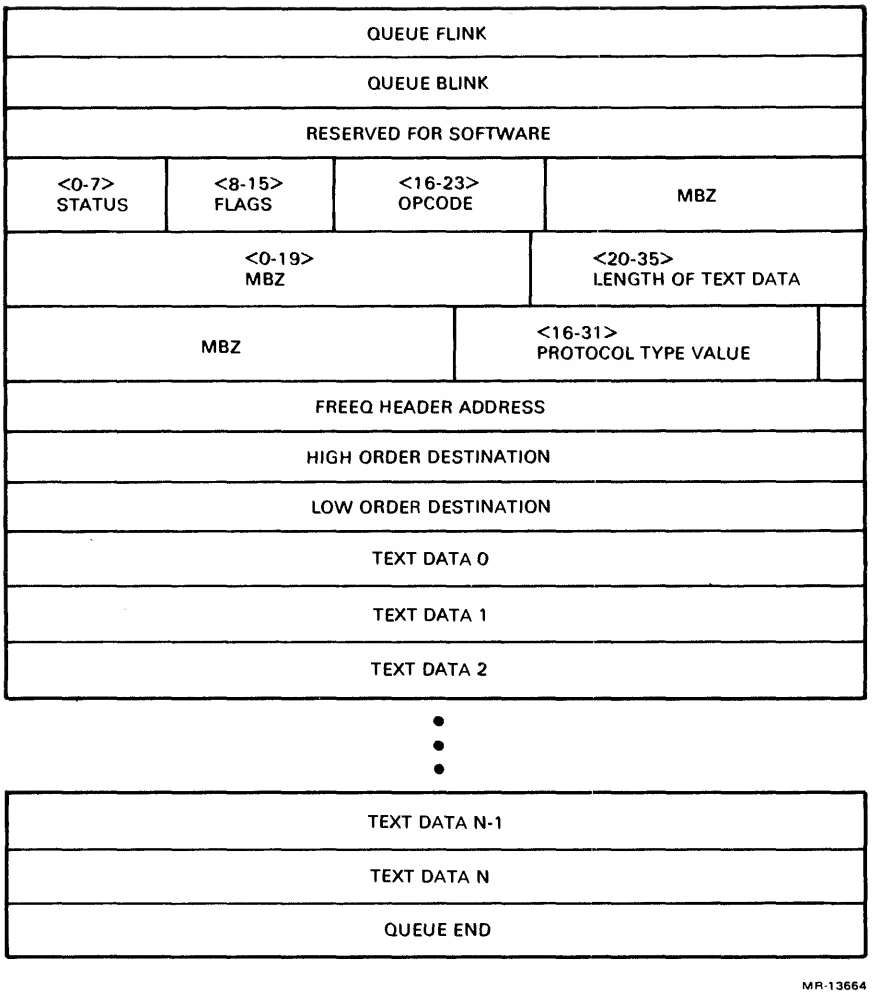


Figure 2-16 SNDDG Command Format (Non-BSD)

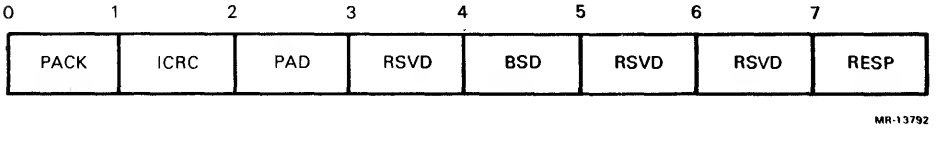


Figure 2-17 Send Datagram

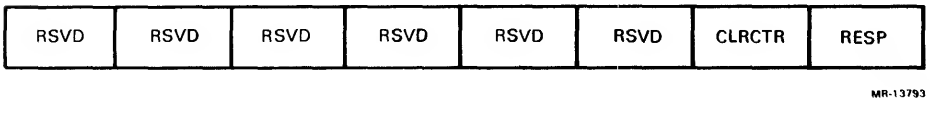


Figure 2-18 Non-Send Datagram

If the BSD bit is 1, the datagram is using the BSD format described under the SNDDG command. If the bit is 0, the datagram is in immediate mode; that is, the data to be transmitted follows the destination address in the queue entries.

Bit 6 is reserved for a send datagram and non-send datagram. This field is valid only for the read counters command. If this bit is set, all the counters will be cleared after their values are reported in the response packet for this command.

When the response bit is 1, the port will always build a response after processing the command.

The FREEQ header address indicates the free queue from which the queue entry was obtained so that, when done, it can be put back in the correct free queue.

The high or low destination words specify the destination address of the packet. Their format is described in Figure 2-19. Note that byte 0 is the first byte transmitted on the NI, and byte 5 is the last byte transmitted. Bit 31 of the low-order destination is therefore the multicast address bit.

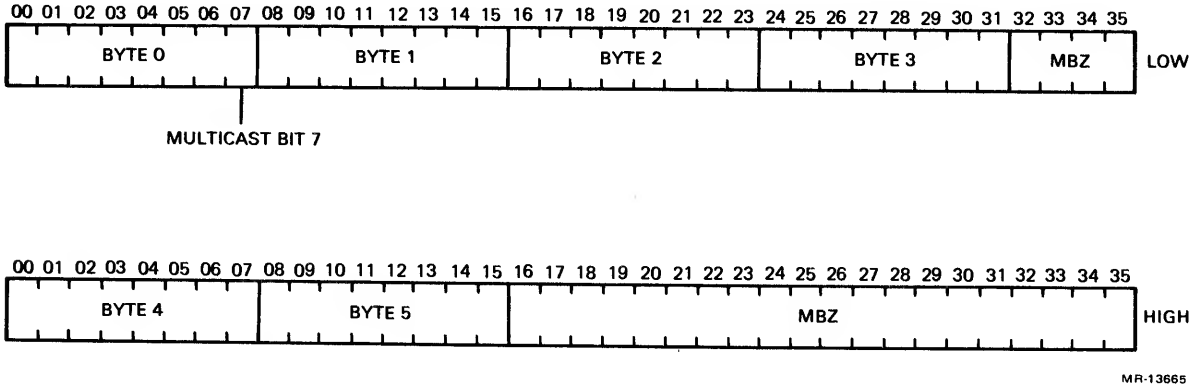


Figure 2-19 Destination Address Format

2.4.1.2 **Datagram Sent (DGSNT) Response** -- A response to the SNDDG command is built if 1) an error occurred during processing of the command, or 2) the response bit of the flags field of the original datagram packet was set. The format of the returned packet is shown in Figure 2-20. The format of the response follows the format of the original command. Thus, if the BSD bit of the flags field was on, then the response is in BSD format as well. Figure 2-20 shows the format for a non-BSD packet:

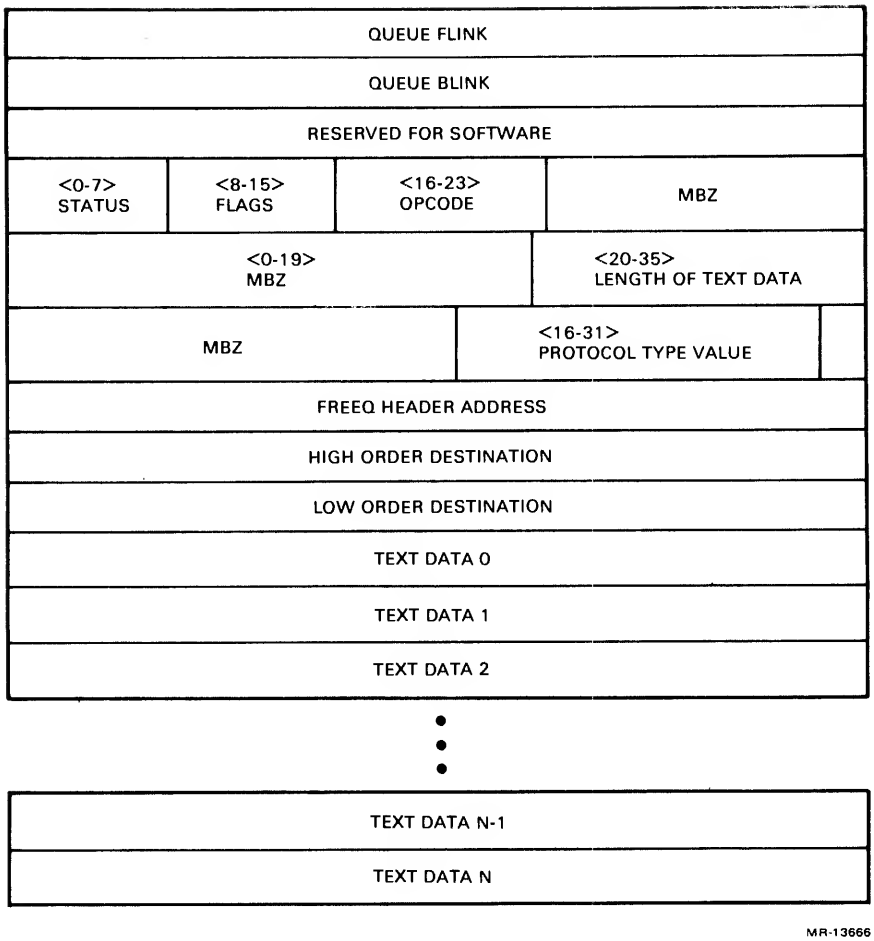


Figure 2-20 DGSNT Response Format (Non-BSD)

The format of a response for a send datagram command packet with the BSD bit of the flags field set is shown in Figure 2-21.

The format of the individual words and the bits therein conforms to the information given for the SNDDG command format.

The status field shown in Figure 2-22 is used by the port to report the status of all completed commands. This field appears in the response word of the queue entry. When valid, this field indicates the logging of an exception event.

When the CRAM PE field is set to 1, the forthcoming read counter response is due to execution of a planned CRAM parity error.

When the send/receive bit is 0, an error occurred on receive; receive failed. When this bit is 1, an error occurred on transmit; transmission failed.

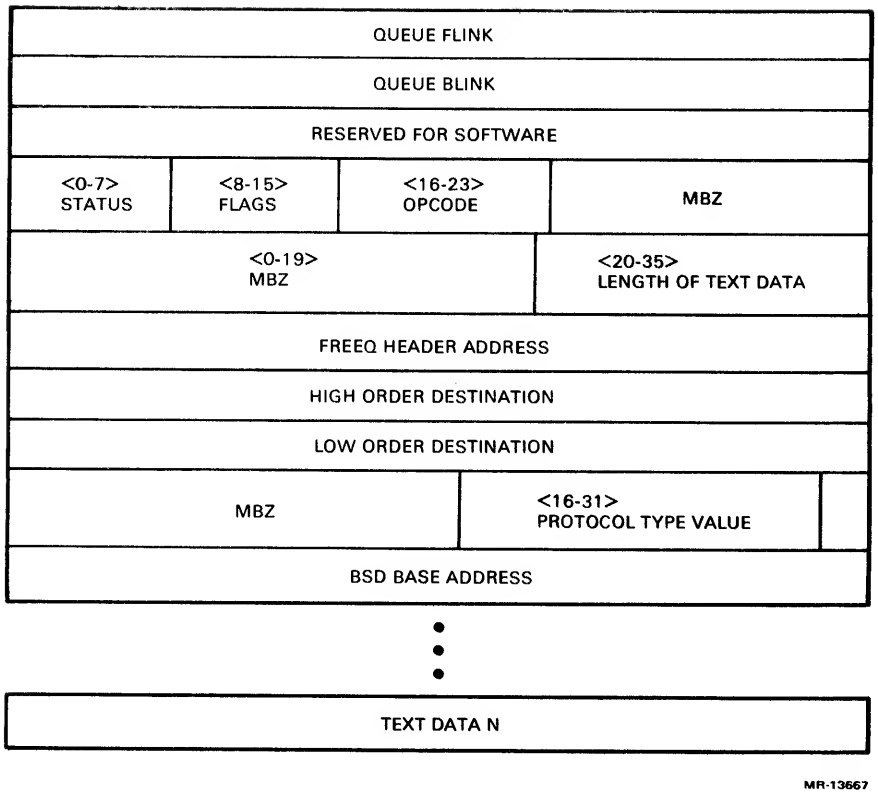


Figure 2-21 DGSNT Response Format (BSD)

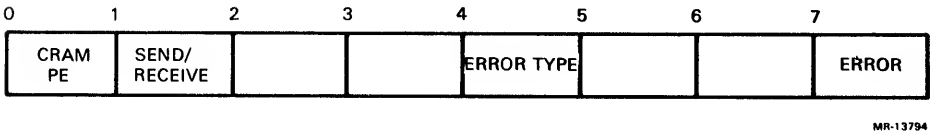


Figure 2-22 Status Field from Queue

When the error bit is 0, the status field has no meaning and must be zero (MBZ). When this bit is 1, the status field is reporting an error event. The definition of the error type fields, and of the direction field, comes into effect.

The error type field indicates the error event being logged. The field is set according to Table 2-2.

2.4.1.3 Datagram Received (DGRCV) Response -- When a datagram is received, a response packet is built. The format of a received datagram is shown in Figure 2-23.

The operation code field for a received datagram is 5.

Table 2-2 Error Log and Type

Bit Value (Octal)	Event Type	Note
00	Excessive collisions	2
01	Carrier check failed (carrier lost)	2
02	Collision detect check failed	
03	Short circuit	4
04	Open circuit	4
05	Frame too long	
06	Remote failure to defer (late collision)	4
07	Block check error (CRC error)	
10	Framing error	
11	Data overrun (NIA buffer space exhausted)	
12	Unrecognized protocol type	
13	Frame too short	3
30	Channel error WC not equal zero	
31	Queue length violation	5
32	Illegal PLI function	
33	Unrecognized command	
34	Buffer length violation	1
35	Reserved	
36	Transmit buffer parity error	
37	Internal error	

NOTES:

1. For a transmission, this error means that the length information in the transmitted BSD was inconsistent, such as when the length field of the transmitted datagram does not match the total length of the BSDs to be transmitted.

2. When this event is being logged, bits 26-35 of the opcode word in the response packet of the queue entry become the time domain reflectometry (TDR) reference number obtained from the NIA when the error occurred. This indicates the time, in 100 ns tics, from the time the transmission started until the error event was detected by the NIA hardware.

3. This error occurs only when padding of the transmitted frame is disabled, and the frame length is smaller than the smallest legal Ethernet frame size of 46 data bytes (64 bytes including all physical channel protocol). No indication is given if the frame size would be a runt, and padding is enabled. In that case, the frame is padded as noted in the description of the PAD flag in Figure 2-17.

4. These errors are also called late collision errors. A late collision error is defined as a collision that occurs after the slot time (51.2 μsec) has expired. The slot time is the maximum length of time for a signal to propagate from one end of an NI network and back. All other nodes on a network should observe that a station is transmitting and they should defer.

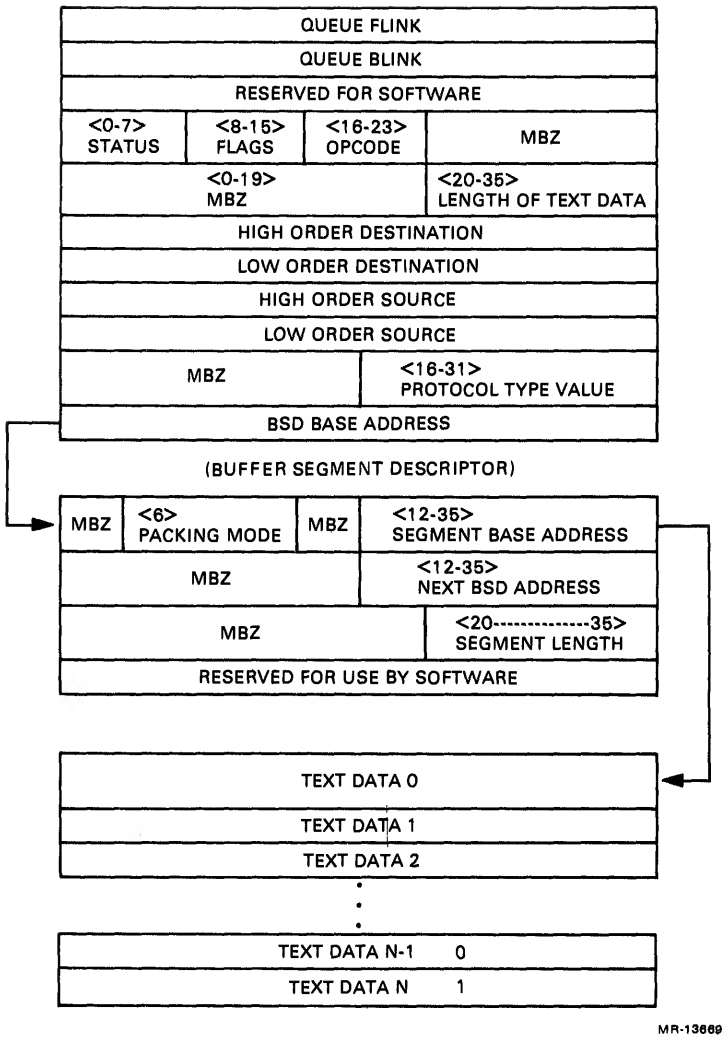


Figure 2-23 DGRCV Response Format

5. This error indicates the host memory free space was exhausted while there was data remaining in the NIA receive buffer.

The text data field contains the length of the transferred text data in bytes, plus four bytes to include the cyclic redundancy check bytes at the end of the packet. A received packet has appended to it the CRC transmitted by the transmitting node. The length value does not include the packet header. This is the length of the data portion of the datagram. The packet length field always indicates the actual number of bytes in the packet, even if the packet is too long, or if an error of some sort occurs. The received CRC bytes are placed into the host memory buffer immediately following the end of memory data to allow a software double-check of packet integrity.

The protocol type field contains the protocol type of the received packet. The format for this field is the same as that for the SNDDG command, and the protocol type field of a PTT entry.

The destination high/low field contains the destination port address as received from the NI wire. This field is included so that messages received for a multicast address (by the port) can be distinguished (by the driver) from messages received for the physical address of the port.

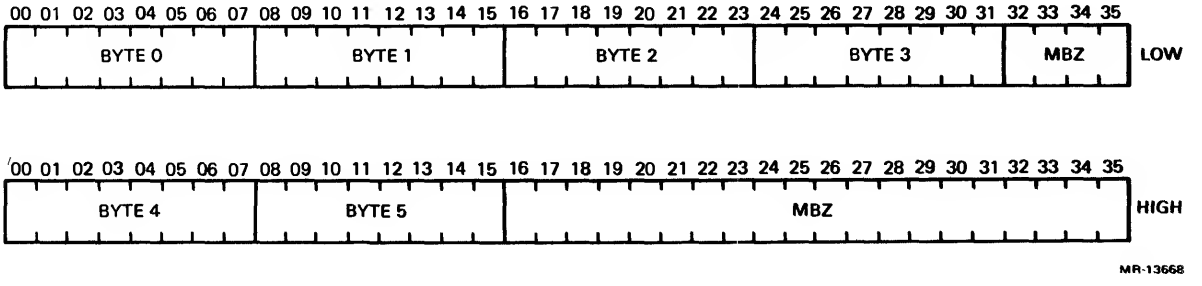


Figure 2-24 Originating Port Address Format

The source high/low field contains the originating port address. The format of this address is given in Figure 2-24, where byte 0 is the first byte received over the NI wire. Byte 5 is the last byte received.

Text data 0 contains the first byte of the received packet. The text portion of a received datagram is always assembled by the port left to right. That is, the first byte of text received will be placed into bits 0-7 of the first text word, the second byte of text received will be placed into bits 8-15, the third into bits 16-23, the fourth into bits 24-31, and so on. This occurs for all data modes.

For a description of the BSD base address refer to Section 2.3.3.1.

2.4.1.4 Load Protocol Type Table (LDPTT) Command -- When this command is accepted by the port, the PTT specified will be cached internally to the port local RAM storage memory. Note that the protocol free queue addresses are cached internally as well. Therefore, the free queue headers cannot change addresses unless this command is issued as well. The addresses specified in this table point to the free queue header, not to the first queue entry of the queue chain.

The address of the PTT specified in the PCB may not be altered in a running port without initialization.

The format of the LDPTT command is specified in Figure 2-25.

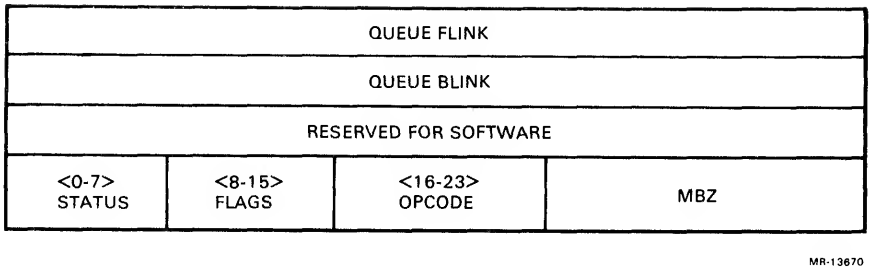


Figure 2-25 LDPTT Command Format

The operation code for this command packet is 3. When the command is accepted by the port, the PTT is internally cached. There is no additional queue data.

If the LDPTT command is not executed before enabling the port after initialization (by setting bit 31 of the CSR), then no protocol types are enabled. Any packets that pass the receive address filter will be linked to the unknown protocol type queue.

2.4.1.5 Protocol Type Table Loaded (PTTLD) Response -- The LDPTT command signals completion by building a response and putting that response at the end of the response queue. The format of that response is given in Figure 2-26.

The driver should not assume that a protocol type has been enabled until this response is received from a LDPTT command. The opcode and flags fields are not modified by this response (same as for the command).

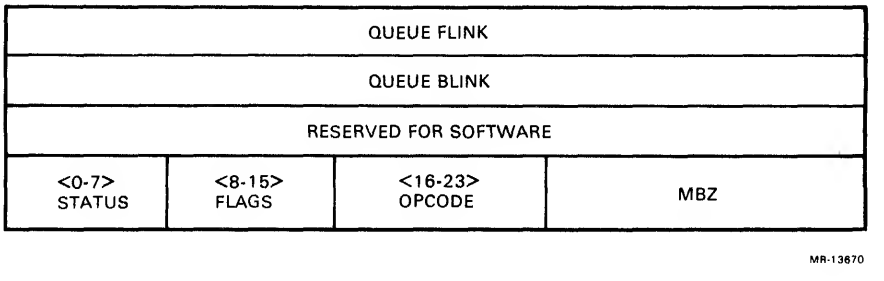


Figure 2-26 PTTLD Response Format

2.4.1.6 **Load Multicast Address Table (LDMCAT) Command** -- When this command is issued, the multicast address table (MCAT) is loaded in the port from the table whose address is specified by the PCB. At initialization, no multicast addresses are enabled to allow upper layers of software (users) to initialize before enabling.

For the driver to add or delete a multicast address to or from a running port, a MCAT is built in memory at the address pointed to by the PCB MCAT base address pointer with the addresses in increasing numerical order. Then the LDMCAT command is issued. When the response to this command is returned, the multicast addresses specified in this table are in effect.

The table is loaded into the internal cache of the port. This table must follow the format stated for the MCAT (see Figure 2-27). This table must be loaded before address checking will take effect. The operation code for this command is 2. There is no other queue data.

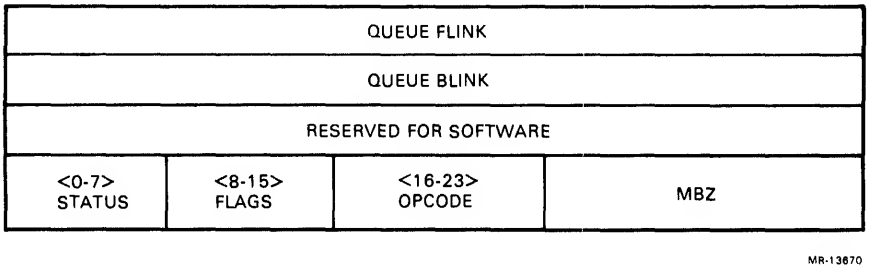


Figure 2-27 LDMCAT Command Format

2.4.1.7 MCAT Loaded Response LDMCAT -- The successful completion of the LDMCAT command is signalled by the building of the following response on the response queue. The format of the response is shown in Figure 2-28. The driver can assume that the

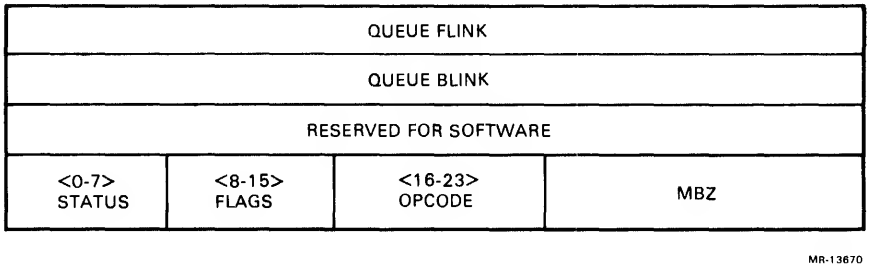


Figure 2-28 MCATLD Response Format

specified MCAT has been loaded only after it has received this response packet. The command opcode and flags fields are not modified by this command.

2.4.1.8 Read and Clear Performance Counters RCCNT Command -- This command reads the performance counters, returning their value in the read counters block pointed to by PCB +30, and clears the performance event counters as specified by a bit (14) in the flags field.

The format of the command queue entry to accomplish this is given in Figure 2-29.

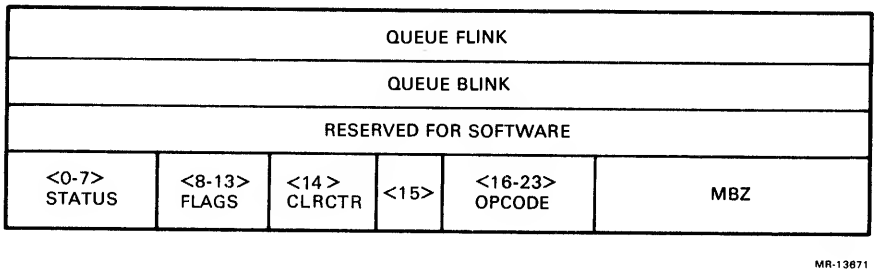


Figure 2-29 RCCNT Command Format

The operation code for this command packet is **4**. A response is built if the response bit in the flags word is set.

Datagram discarded counters are kept for each of the protocol type queues, and for the unknown protocol type queue. Thus, the driver can get an indication when a protocol type free queue is exhausted by execution of this command. A counter is returned for every protocol type entry allowed in the PTT. Those counters corresponding to protocol type entries that are not enabled are returned with a value of zero.

2.4.1.9 Counters Read or Cleared (CNTRC) Response -- If the response bit of the flags field of the original command packet is on, then a response to the above command is built. The response has the format shown in Figure 2-30.

By combining the read and clear commands, the driver knows that the event counters are not skewed when a clear command is issued. In the following list, note that a packet is not received unless it passes the receive address filter.

BR -- Bytes received. This counter represents the number of 8-bit data text characters received as datagrams over the NI. This includes maintenance operation protocol (MOP) packets, explained in section 2.6.

BX -- Bytes transmitted. This counter represents the number of 8 bits data text characters transmitted successfully as datagrams over the NI.

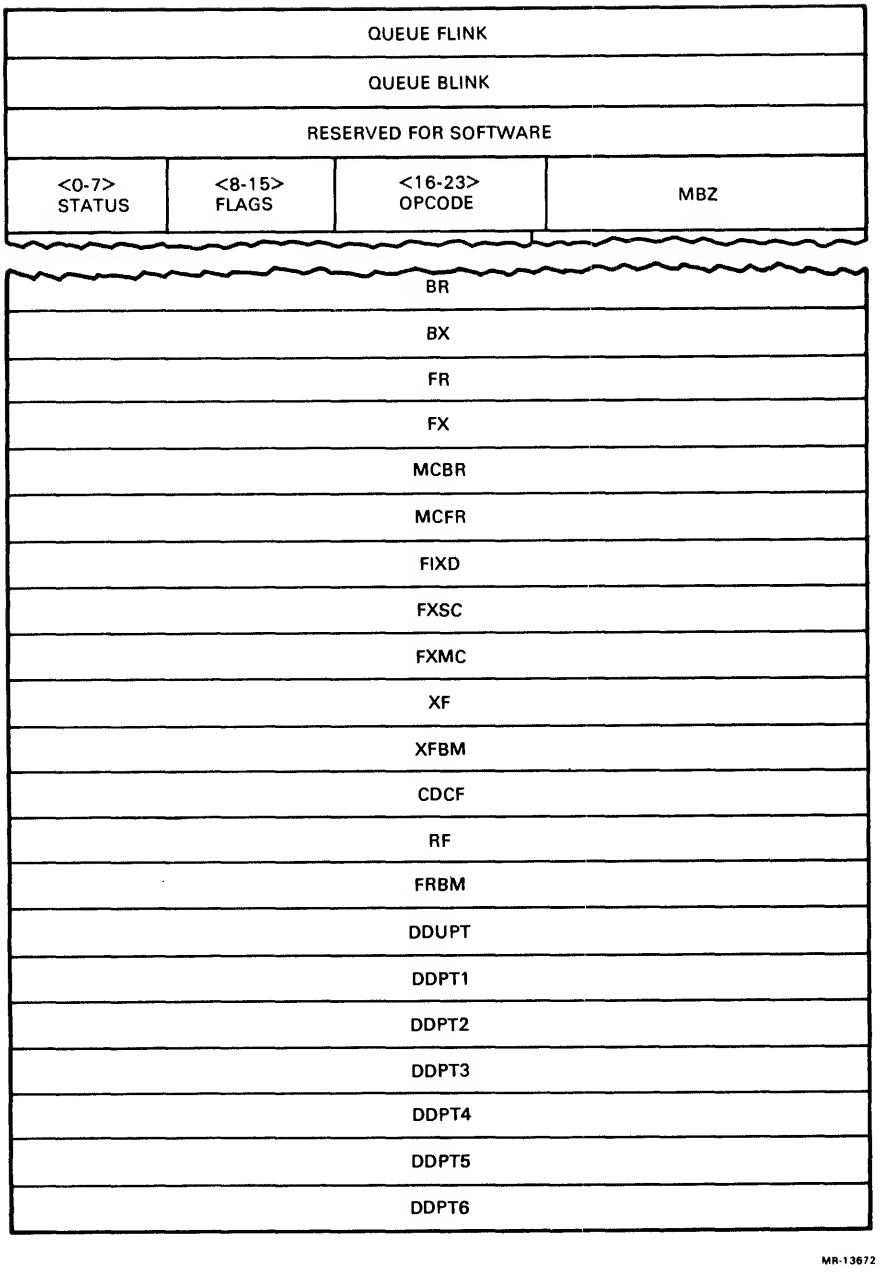


Figure 2-30 CNTCL Response Format
(Sheet 1 of 2 sheets)

FR -- Frames received. This counter represents the number of frames (packets or datagrams) that have been received over the NI wire.

FX -- Frames transmitted. This counter represents the number of frames that have been successfully transmitted over the NI wire.

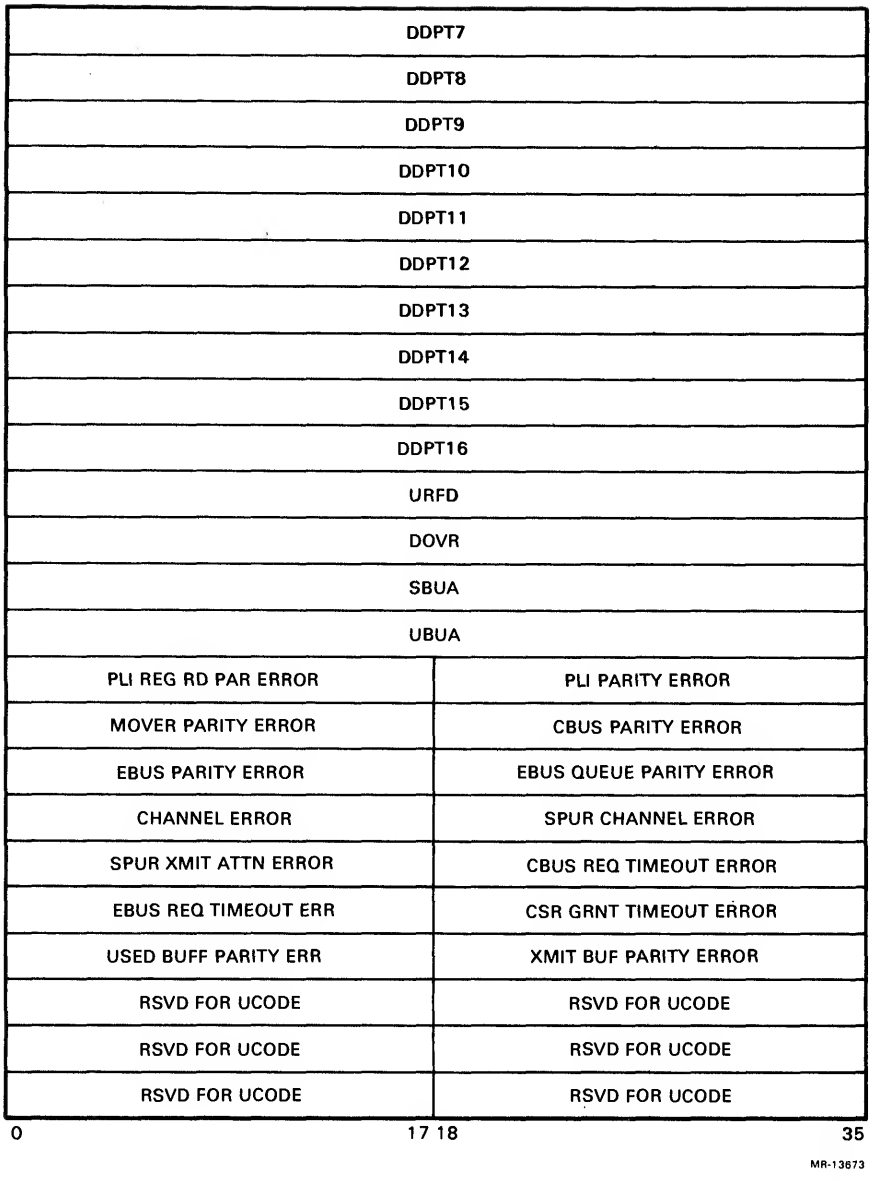


Figure 2-30 CNTCL Response Format
(Sheet 2 of 2 sheets)

MCBR -- Multicast bytes received. This counter represents the number of 8-bit bytes received in packets with the multicast bit set in the destination field. This includes broadcast.

MCFR -- Multicast frames received. This counter represents the number of frames that were received with the multicast bit in the destination field set. This includes broadcast.

FXID -- Frames transmitted, initially deferred. This counter represents the number of frames transmitted that had to defer to other traffic on the NI wire before transmission.

FXSC -- Frames transmitted, single collision. This counter represents the number of frames that were successfully transmitted, and which collided with another transmission exactly once.

FXMC -- Frames transmitted, multiple collisions. This counter represents the number of frames that were successfully transmitted, and which collided with another transmission more than once.

XF -- Transmit failures. This counter represents the number of frames that were not successfully transmitted. This counter is incremented for excessive collisions, parity errors, and so on. This counter is associated with the XFBM, which notes occurrence of error classes.

XFBM -- Transmit failure bit mask. This counter gives the accumulated reasons for transmission failures. The bit meanings are given in Table 2-3.

Table 2-3 Transmission Failure Bit Mask Assignments

Bit No.	Reason for Failure
0-23	Unassigned
24	Loss of carrier
25	Transmit buffer parity error
26	Remote failure to defer
27	Frame too long
28	Open circuit
29	Short circuit
30	Carrier check failed (collision detect check failed)
31	Excessive collisions

CDCF -- Collision detect check failed. This counter gives the number of times that the collision detect check failed after a transmit. This is the number of times that heartbeat failed to assert after a transmit ended. This counter has meaning only if the H4000 mode bit is set.

RF -- Receive failures. This counter gives the number of received frames whose reception ultimately failed. This counter is associated with the RFBM counter, which marks occurrence of the various error types.

RFBM -- Receive failure bit mask. This counter gives the accumulated reasons for receive failures. The bit definitions are given in Table 2-4.

Table 2-4 Reception Failure Bit Mask Assignments

Bit No.	Reason for Failure
0-26	Unassigned
27	Free list parity error
28	Data overrun (no free buffers)
29	Frame too long
30	Framing error
31	Block check error

DDUPT -- Datagram discarded for unknown protocol type. This counter keeps track of the number of datagrams discarded for the unknown protocol type free queue. Any time a datagram is discarded with an unrecognized protocol type, this counter is incremented.

DDPT1 to DDPT16 -- Datagram discarded for protocol type N. These counters keep track of the number of datagrams discarded for each of the protocol type free queues. When a datagram is discarded because of no available free space, one of these counters is incremented, if the protocol type was enabled. There are as many of these counters as needed to support the number of protocol types allowed in the NI configuration register.

URFD -- Unrecognized frame destination. This counter has no meaning for the NIA20 and will always be reported as zero.

DOVR -- Data overrun. This counter represents the number of packets incorrectly received due to buffer space in the NIA being exhausted. Such packets are discarded.

SBUA -- System buffer unavailable. This counter has no meaning for the NIA and will always be reported as zero.

UBUA -- User buffer unavailable. This counter represents the total number of packets discarded because a free queue was exhausted. This number is the total of 1) the datagram discarded for protocol type N counters and 2) the datagram discarded for unknown protocol type counter.

Words 47-55 are the NIA20 port recoverable errors. These counters have an initial threshold of 5, set during port initialization. This threshold count is variable and can be set via a write station information command.

RSVD -- Reserved For microcode. These counters are reserved for the port microcode and for the present will be returned as zeros.

2.4.1.10 Write Port Link Interface (WRTPLI) Command -- This command causes a PLI write cycle for the indicated operation to be performed.

CAUTION

Execution of illegal or random commands will compromise the functionality of the interface in an indeterminate fashion. It is recommended that the port not be executing meaningful commands when this command is issued. This command is for diagnostic purposes only.

The WRTPLI command format is shown in Figure 2-31.

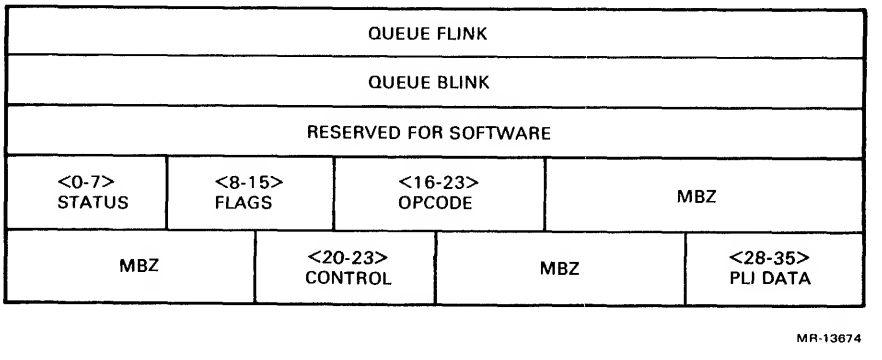


Figure 2-31 WRTPLI Command Format

The operation code for this packet is 6. If the response bit in the flags word is on, a response confirming the correct execution of this command will be built and placed on the response queue anchored in the PCB. This command, along with RDPLI, gives the port driver the same visibility as the port itself into the port link interface.

The control field specifies the PLI control bits to be put onto the PLI when the write cycle is done. The PLI data field is light bits wide. The use of these control bits will be explained in Chapters 4 and 5. The command codes map into the PLI commands as shown in Table 2-5.

Table 2-5 Command Code Values and Functions

Value (octal)	Function
00	Illegal
01	Receive buffer to transmit buffer
02	Write free buffer list
03	Read receive buffer
04	Write transmit action (4 command group)
05	Reset receive attention
06	Enable link control
07	Disable link control
10	Write address register
11	Write transmit buffer
12	Write register
13	Clear receive buffer

2.4.1.11 Port Link Interface Written (PLIWRT) Response -- The WRTPLI command results in a response if the response bit of the flags field of the original packet was set. The driver is assured that the port interface has executed this command only after the response has been received. If built, the response has the format shown in Figure 2-32.

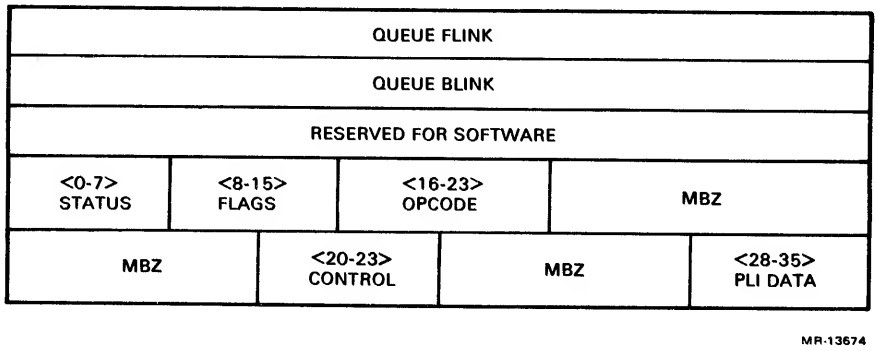


Figure 2-32 PLIWRT Response Format

2.4.1.12 Read Port Link Interface (RDPLI) Command -- When executed, this command causes a read PLI cycle to be executed, using the control bits specified. The data is returned in the response queue entry.

The data for this command is returned in the response entry built if the response flag is set when the command is executed.

The format of this command is specified in Figure 2-33.

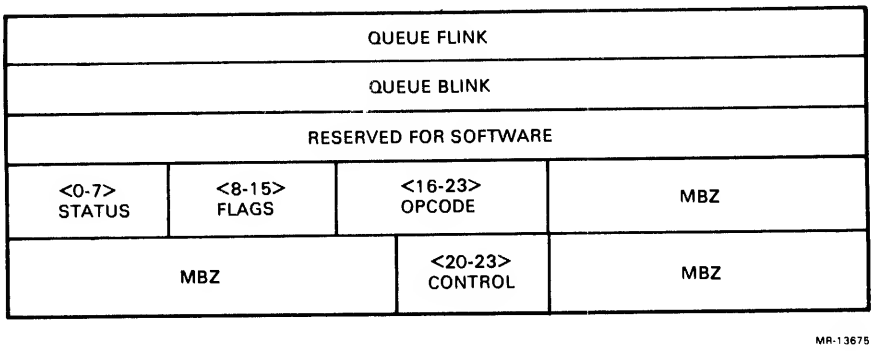


Figure 2-33 RDPLI Command Format

2.4.1.13 Port Link Interface Read (PLIRD) Response -- The format of the response given to this command if the flags response bit is on is specified in Figure 2-34. The operation code for this packet is 7. This command does not generate an NI packet. Through this command, the port driver can read any quantity that the port processor itself can read over the PLI.

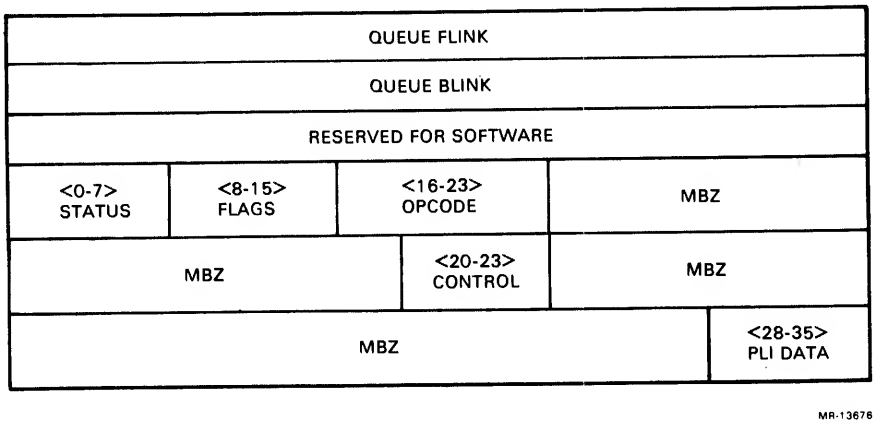


Figure 2-34 PLIRD Response Format

The four control bits specify the state of the control bits to be used during the read cycle. The use of these control bits will be explained in Chapter 4. The command codes map into PLI commands as defined in the Table 2-6.

Table 2-6 Control Bit Values and Functions

Value (octal)	Function
00	Illegal
01	Read register
02	Read receive buffer
03	Read used buffer list
04	Read transmit status
05	Read receive status

If the command selected is not defined in Table 2-6 as a legal function, a response will be generated with the status field set illegal PLI function.

The PLI data field (and the word that contains it) is present only in the response built for this command -- occurring when the response bit in the flags field is set. In the response, this field is the data that was read from the PLI by the execution of the RDPLI command.

CAUTION

It is recommended that the driver not execute this command while other commands are pending or while packet reception is enabled. Careless use of this command can cause the normal port functionality to be compromised.

2.4.1.14 Read NI Station Address (RDNSA) Command -- When executed, this command causes the NI station address to be read from the NI link module. In addition, some status mode bits are read. This data is returned in the response queue entry, if specified by the response bit in the flags field of the command.

The format of this command is specified in Figure 2-35. The operation code for this packet is 8. If the response bit in the flags word is on, a response confirming the correct execution of the command will be built and placed onto the response queue anchored in the PCB.

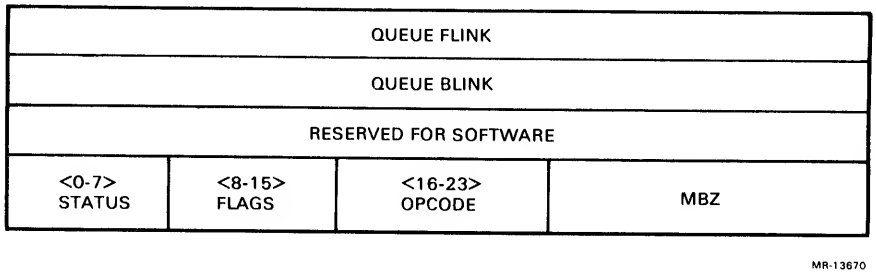


Figure 2-35 RDNSA Command Format

2.4.1.15 NI Station Address Read (NSARD) Response -- If built, the response to the RDNSA command is shown in Figure 2-36.

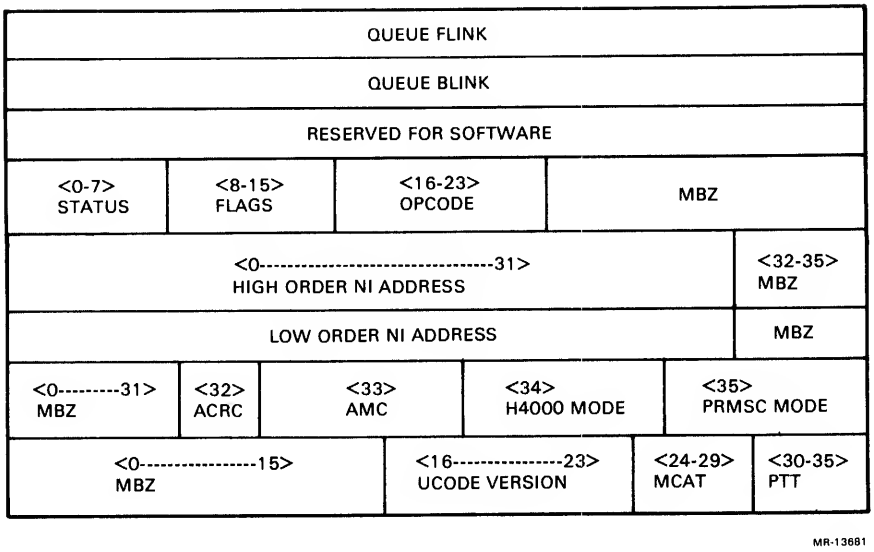


Figure 2-36 RDNSA Response Format

The high and low order NI address value is the physical station address stored in the physical address RAM on the NI link board. It corresponds to the address of the NI link.

The ACRC bit indicates whether the NI link will discard incoming packets with CRC errors. If set, the link will accept all incoming packets with CRC errors and place them on the unknown protocol type free queue if an entry is available. If reset, the link will discard all incoming packets with CRC errors. The initial value of this bit is 0.

The AMC bit indicates whether the NI link will accept all multicast packets (if set) or perform normal multicast address filtering (if reset). The initialization value of this bit is 0.

The H4000 mode bit shows its current state. This bit, if set, enables the heartbeat detection checking for the H4000 type NI bus tranceiver. The initialization value of this bit is 0.

The PRMSC mode bit, if set, indicates that the link is operating in promiscuous mode. All packets detected on the wire will be interpreted as being addressed to this node. The initialization value of this bit is 0.

The Ucode version field gives the version of microcode loaded into the port.

The MCAT field gives the number of MCAT entries allowed.

The PTT field gives the number of PTT entries allowed.

2.4.1.16 Write NI Station Address (WRTNSA) Command -- When executed, this command sets the physical address RAM on the NI link board, as well as the several mode bits for the data link operation.

A response for this command is built only if the response bit in the command packet flags byte is on. The format of this command is illustrated in Figure 2-37. The operation code for this command is 9.

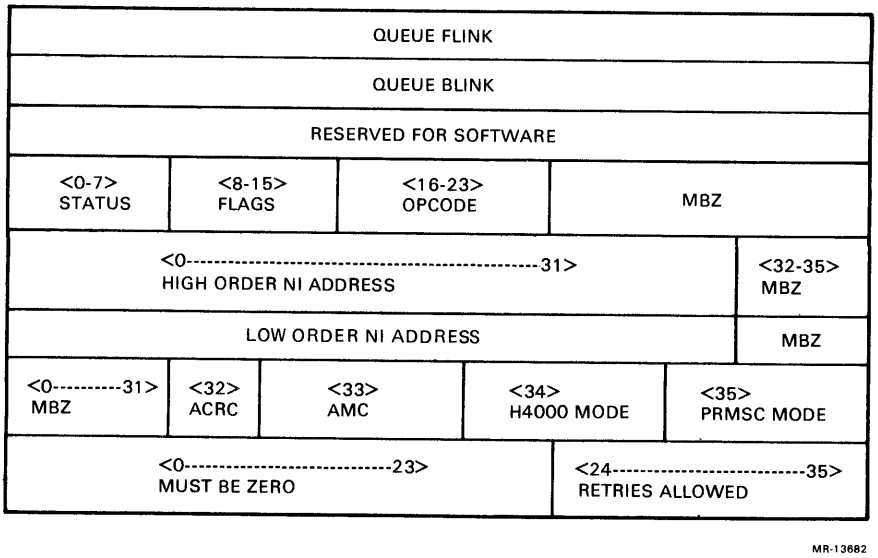


Figure 2-37 WRTNSA Command Format

The high/low order NI address is written to the physical address RAM on the NI link board. After the command is completed, the NI link will accept packets with the physical address specified.

The ACRC bit indicates whether the NI link will discard incoming packets with CRC errors. If set, the link will accept all incoming packets with CRC errors and place them on the unknown protocol type free queue if an entry is available. If reset, the link will discard all incoming packets with CRC errors. The initialization value of this bit is 0.

Setting the AMC bit puts the port into the receive all multicast mode. When enabled, all multicast packets received are passed by the receive address filter.

The H4000 mode bit, if set by the driver, will enable H4000 mode -- enable heartbeat detection checking by the transceiver.

If PRMSC mode is set by the driver, all packets seen on the NI cable will pass the received address filter. This is a diagnostic feature.

CAUTION

Turning on this mode can cause significant degradation of system network performance, depending upon network load.

The retries allowed field specifies how many retries of retrievable errors are to be attempted by the port before the error is declared uncorrectable. The default number of retries is 3.

2.4.1.17 NI Station Address Written (NSAWRT) Response -- The response format to the WETSNA command is given in Figure 2-38. The definitions of the bits for this response are the same as those for the WRTNSA command.

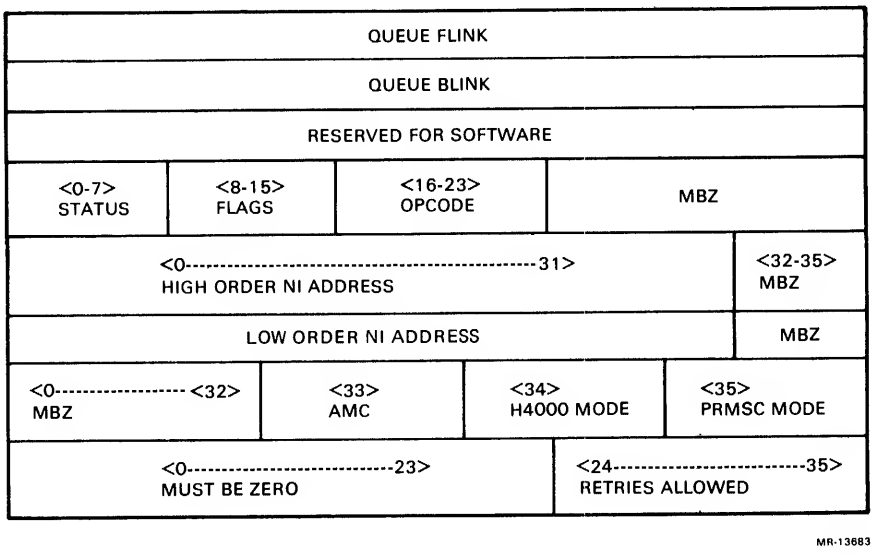


Figure 2-38 NSAWRT Response Format

2.4.2 Self-Directed Commands -- Loopback

It is specifically allowed for a datagram transmitted to be destined for the transmitting node, as one form of loopback. The design of the NI adapter (the NI physical channel), however, shares the CRC generator/checker between the receive and transmit circuits.

Since the CRC circuits must be used to check the incoming packet, packets destined for the transmitting node must supply a CRC code that will be transmitted in place of the internally generated CRC.

This driver CRC is appended onto the end of the normal data packet, and the included CRC (ICRC) bit of the packet FLAGS field is set. The text length does not include the appended CRC.

2.5 DATA FORMATTING/PACKING MODE

The KL10 NI port supports one data formatting mode. The mode is industry compatible. Figure 2-39 illustrates the industry-compatible mode for mapping 8-bit NI bytes into 36-bit KL10 words.

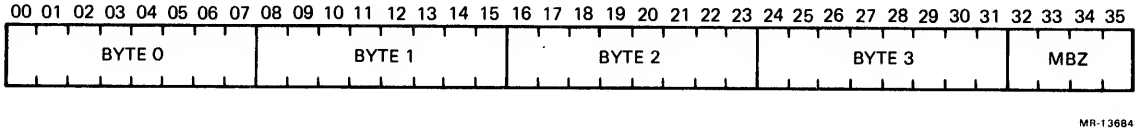


Figure 2-39 KL10 Word (Industry-Compatible Mode)

2.6 MAINTENANCE OPERATION PROTOCOL

The network management specification requires that each NI node handle certain maintenance operation protocol (MOP) messages:

1. Request ID (REQID)
2. Loopback (LPBK)
3. Read counters (RDCNT)

In addition, the Ethernet specification includes these client layer protocols. Microcode space considerations do not allow these features to be implemented in the NIA20 port microcode.

Higher layers of software will handle the MOP messages; in fact, the port microprocessor should not be giving correct responses to MOP messages if these higher layers of software are not present and functioning.

2.7 ERROR HANDLING

The port microcode is capable of retrying many operations that fail -- to allow a large part of the error recovery to be built into the port microcode and removed from the port driver. When the port encounters a fatal, nonrecoverable error, the port will stop processing any more commands. The port will then move the command queue entry to the tail of the response queue and request an interrupt. The port will then enter the disabled state.

While the port is waiting for port driver intervention, it will not be emptying packets out of the receiver buffer, so packets may be discarded due to lack of available buffer space. While the port is in the disabled state, no increment of a datagrams discarded event counter is made.

A class of more severe errors causes the port to cease operations immediately and exit to a special microcode location that has a CRAM parity error. The host will notice the error location, and will reinitialize the port, if possible. Errors in this class indicate a failure in the interfaces between the port and the external world (e.g., memory, the EBus, the CBus, interrupt structure). Errors of this severity indicate a serious port hardware problem. In general, the port is trying to indicate that it cannot communicate with anything.

2.7.1 Error Events

Some errors may not necessarily involve a hardware malfunction. Such errors include excessive collisions, CRC errors, and framing errors. The occurrence of these errors terminates the packet transmission or reception in progress when the error occurs, but does not affect the processing of other commands or packets being received.

Errors of this sort are reported by creating a response packet for the transfer involved, and setting the status field accordingly. Table 2-7 provides a list of possible error conditions.

2.7.2 Discarded Datagrams

Under certain conditions, received datagrams may be discarded because of insufficient buffer space -- characteristic of the datagram class service offered by the NI port. For example, if a received datagram is occupying buffer space in the NIA and obtaining a free queue entry from some free queue fails, the datagram involved is discarded, and the datagrams discarded counter for the appropriate free queue is incremented. This occurs even if buffer space in the link is still available.

The reasons for this are that 1) other packets being received may be able to be stored since the free queue error condition is protocol-type-dependent, and 2) if the link fills up, datagrams will be discarded without increment of a datagrams discarded event counter. No other error indication or response is made. It is up to higher levels of the network to detect and recover from such errors.

If the port is addressed by a burst of packets such that the internal buffer space in the NI physical channel is exhausted, datagrams may be discarded without increment of the datagrams discarded counter. The buffer space allowed in the physical channel (NI link) is sufficient to make this event unlikely. The NIA has 16K bytes of buffering, normally organized as 32 pages of 512 bytes each. This storage is sufficient for 32 minimum-size packets and 10 maximum size packets.

Table 2-7 Possible Error Conditions

Error Condition	Description
Unrecognized command	Packet (from driver) has invalid operation code
Buffer length violation	BSD has inconsistent buffer length information
CRC error	Received packet has CRC error
Framing error	Received data not byte aligned (not an integral number of octets in frame (number of bits not evenly divisible by 8)
Packet too long	Packet length exceeds maximum Ethernet packet length
Excessive collisions	Packet transmission attempt collided 16 times in succession
Carrier lost	Carrier lost before end of packet detected
Collision detect check	Collision detect heartbeat failed to assert
Remote failure to defer (also called late collision)	Collision occurred after the slot time expired
System buffer unavailable	Port microcode was unable to get a free queue in KL10 memory

2.7.3 Event Counters

To provide for performance measurement, and for diagnostic purposes, a number of event counters are provided by the port. These record the occurrence of certain events, and can be read and cleared upon command by the driver program. The read and clear counters (RCCNT) command reads the value of these counters and returns them in the read counters block.

These event counters do not necessarily record, or imply, abnormal errors, but high counting rates for some of them may point to a hardware or software failure. A list of applicable events is as follows:

1. Received a byte
2. Transmitted a byte

3. Received a frame
4. Transmitted a frame
5. Received a multicast byte
6. Received a multicast frame
7. Transmitted a frame that was initially deferred
8. Transmitted a frame with a single collision
9. Transmitted a frame with multiple collisions
10. Failed to transmit a frame successfully
11. Sent failure reason bit mask
12. Transmitted a frame with a late collision
13. Failed to receive a frame successfully
14. Received failure reason bit mask
15. Discarded a datagram for unknown protocol type free queue
16. Discarded a datagram for protocol type entry 1 free queue
17. Discarded a datagram for protocol type entry 2 free queue
18. Discarded a datagram for protocol type entry 2 free queue through 16 free queue

2.8 CONTROL AND STATUS REGISTER

The control and status register (CSR) is another mechanism for communication between the KL10 and the NIA20 port. It is a 36-bit register that resides in the port's EBus interface module.

The KL10 accesses the CSR by executing CONO and CONI commands and the port accesses the CSR by executing MPLOADCSR and MPREADCSR commands to load or read the register. The register is read/write interlocked to prevent simultaneous access.

A complete explanation of the register bits and their meaning is provided in Chapter 5. It is important here to understand that this method of communication exists. Some of the functions performed using the CSR include:

1. Port is enabled or disabled
2. Port is in the idle loop
3. Microprocessor is in the run state
4. Port requested an interrupt
5. Port error: port informs driver that an error occurred
6. CRAM parity error exists
7. MBus error exists
8. EBus parity error exists

The CSR is important in error handling and reporting. Some classes of errors are severe and are reportable only through the use of the CSR -- for example, a planned CRAM parity error, which is the result of the port executing a known bad parity microinstruction. These instructions are purposely written to the CRAM with bad parity and are used only when certain error conditions appear. It will be detected by the CRAM parity logic and will force a nonvectored interrupt.

2.9 NETWORK ARCHITECTURE AND FUNCTIONAL LAYERS

Networking systems are designed and constructed in functional layers. Each layer performs a specific set of functions and services. The combination of the layers creates what is called the network architecture, where the layers interact to provide total, end-to-end network operation.

Most network architecture conforms to the International Standards Organization's model for Open Systems Interconnection, and the Digital Network Architecture (DNA) is no exception. Figure 2-40 shows how the functional layers of the ISO model and the DNA layers correspond.

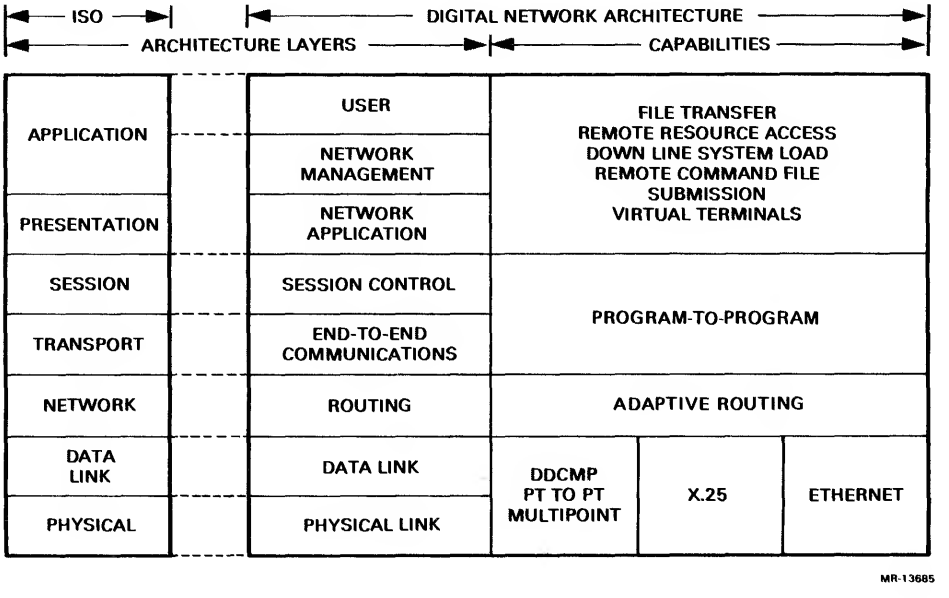


Figure 2-40 Digital Network Architecture (DNA) -- Functional Layers

2.9.1 Physical Link Layer and Data Link Layers

The physical layer is the bottom and most basic layer. It manages the physical transmission of information over the channel. The data link layer, residing immediately above the physical link layer, creates a communications path among adjacent nodes. It frames messages for transmission on the channel connecting the nodes, checks the integrity of received messages, and manages the use of channel resources.

The physical and data link layers operate together to provide a packet delivery (or datagram) service between nodes in the network, where packets are sent from the doorstep of one node to the doorstep of another. The Ethernet specification deals with these two layers and describes the necessary parameters and protocols.

2.9.2 Routing Layer

The routing layer provides a message delivery service, routing user data packets to their destinations. The routing layer and the remaining upper layers support multiple types of data link circuits. In addition to Ethernet, for example, point-to-point and X.25 (packet switched) virtual circuits could be supported.

2.9.3 End-to-End Communications Layer

The end-to-end communications layer provides a system-independent program-to-program communication service. It allows two processes to exchange data reliably and sequentially, independent of which network systems are communicating or their location in the network. Network services protocol (NSP) is used for end-to-end control of addressing, data integrity checking, transaction flows, interrupts, and flow control among communicating processes.

2.9.4 Session Control Layer

This layer provides system-dependent, program-to-program communications functions that bridge the gap between the previous layer and the logical link functions required by processes running under an operating system.

Session control functions include mapping node names to node addresses, identifying end users, activating or creating processes, and validating incoming connect requests.

2.9.5 Network Application Layer

The network application layer controls the network functions used by the two higher layers of DNA. Services include remote file access, remote file transfer, remote interactive terminal access, gateway access to non-DNA systems, and resource managing programs.

2.9.6 Network Management Layer

The network management layer is the only layer that has direct access to each lower layer. It provides the functions to plan, control, and maintain the operation of the network. Functions include down-line loading, up-line dumping, remote system control, test functions, and event logging functions.

2.9.7 User Layer

The user layer contains most user-supplied functions, including programs that access the network and those network services that directly support user and application tasks.

Some examples of the more common services accessed by users include resource sharing, file transfers, remote file access, database management, and network management.

2.9.8 Layer Interfaces

Figure 2-41 shows the relationships among the DNA layers. The top three layers each have a direct interface with the session control layer for logical link services.

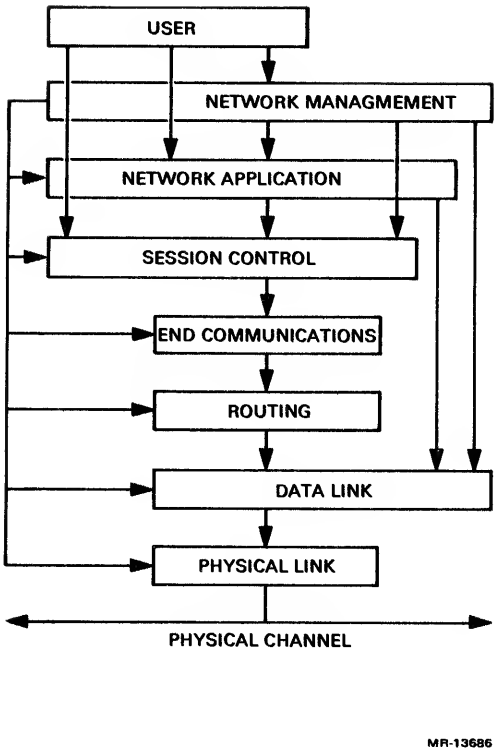


Figure 2-41 DNA Layers and Interfaces

Each layer interfaces with the layer directly below to use its services. The network management layer interfaces to every other layer to get the data needed to control and manage the network.

Horizontal arrows show direct access for control and communication of network parameters. Vertical arrows show interfaces between layers for normal user operations such as file access and down-line loading.

2.9.9 Expanded Ethernet Networks

Extending the capabilities of Ethernet networks can be accomplished by adding communication server products. These products include:

- Terminal and print servers, which connect clusters of terminals and unit record equipment to an Ethernet network

- Router servers, which connect DECnet systems on an Ethernet with remote DECnet systems on another Ethernet or other DECnet network. Router servers do not need to convert protocols, since they enable nodes of like architecture to communicate

- Gateway Servers

  1. Systems network architecture (SNA) gateway servers, which connect DECnet systems with an IBM SNA network

  2. X.25 Gateway servers, which enable an X.25 public packet-switched network to connect DECnet systems with remote DIGITAL or non-DIGITAL systems

Gateway servers can translate protocols to allow communication between nodes with different network architectures.

Figure 2-42 shows a DECnet with many connected Ethernet segments.

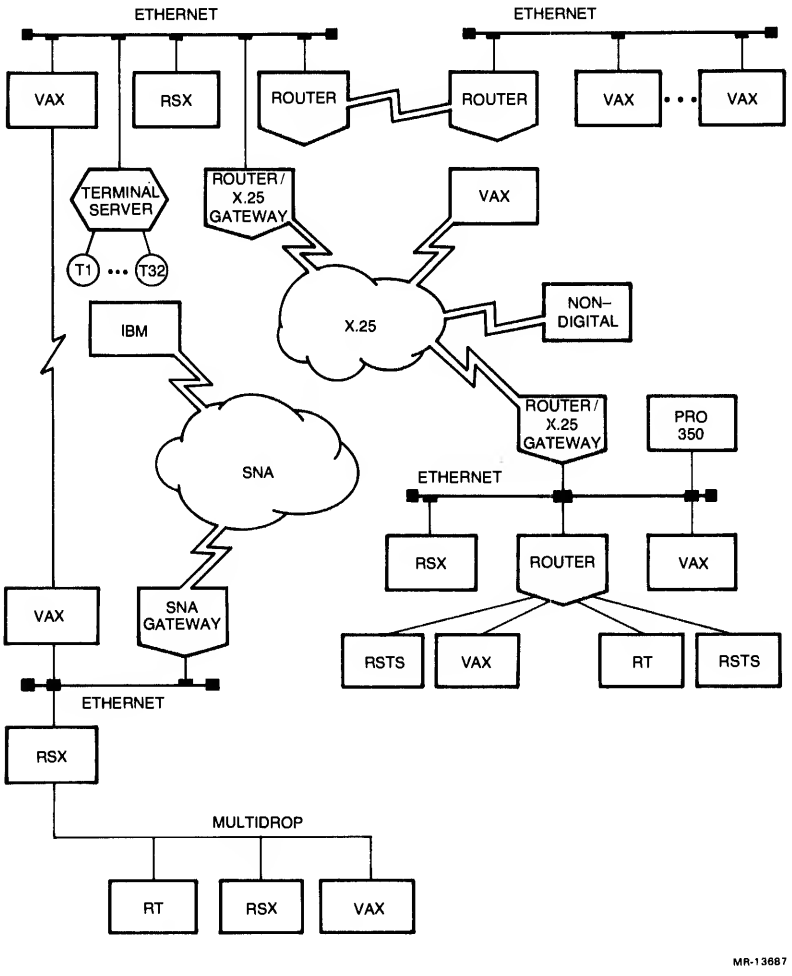


Figure 2-42 DECnet Network with Many Ethernet Segments

CHAPTER 3
INSTALLATION OF NIA20 IN KL10-E

3.1 OVERVIEW

This chapter describes the installation of the NIA20 Network Interconnect Adapter in a KL10-E system. Appendix A describes the installation of the NIA20 in a KL10-D. Appendix B describes the installation of the NIA20 in a KL10-R. Figure 3-1 and Figure 3-2 show the NIA20 installed in a KL10-E, rear and front views, respectively. Table 3-1 itemizes the NIA20 parts, and Table 3-2 lists the harness and cable connections used in the NIA20/KL10-E installation.

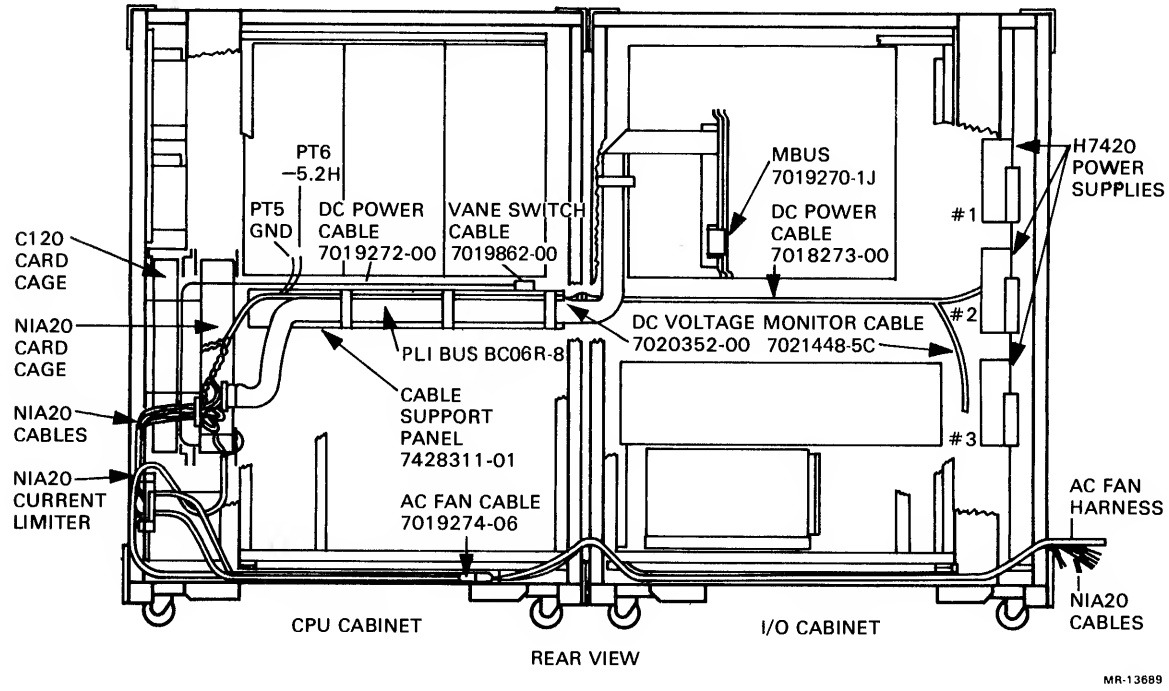


Figure 3-1 NIA20 in KL10-E, Rear View

Table 3-1 Parts List, NIA20 in KL10-E

Line Item	Part No.	Description	Qty
1	7019268-00	Card Cage Assy IPA-20-L	1
2	7019268-01	Card Cage Assy CI20	1
3	7428312-01	Bracket, Interface	1
4	7428222-01	Baffle, Air	1
6	9006073-01	Screw, Mach Pan Phil 10-	13
7	9006022-01	Screw, Mach Pan Phil 6-	6
8	9006633-00	Washer, Lock Internal Steel	6

Table 3-1 Parts List, NIA20 in KL10-E (Cont)

Line Item	Part No.	Description	Qty
9	1213716-00	Spacer, Foam Polyu 1/2	4
10	7020539-06	Cable, Fan AC	1
11	7019274-06	Cable, Fan AC	1
12	7019272-00	Harness, DC-5.2 Sect N1-1 DC+5	1
13	7019273-00	Harness, DC-5.2 Sect N1-2 DC+5	1
14	7019893-2L	Cable Assy Ethernet	1
15	BC06R	BC06R I/O Cable	1
16	7019266-00	Module Blank Assy	1
17	M3002-00	CI20 Microprocessor, Multiwire HE	1
18	M3003-00	CI20 C-Bus/PLI Interface, Multiwire	1
19	M3001-00	CI20 E-Bus Interface, Multiwire HE	1
20	9007032-00	Tie, Cable Bundl. Dia 0-1-3/4"=101	A/R
21	1213715-00	Clip, Flat Cable W/Adhesive Bk	4
22	H7440-00	POA1 H7440	1
23	L0072-00	NI20 (KL10 to NI Adaptor)	1
24	7014103-00	Blank Module Assy	1
25	9107673-06	Pwr Cord, Term 3-14 SJT 115	1
26	7011432-02	Pow Cord Extension 50Hz	1
27	9007651-00	Washer, Lock External Steel	14
28	9006664-00	Washer, Flat SST	12
29	7020352-00	Harness, DC Voltage Monitor	1
30	7019862-00	Harness, Vane Switch	1
31	5414506-01	Voltage, Monitor Board	1
32	7019270-1J	Bus, Cable, M Assy	1
33	3621499-01	Label, DCV Monitor CI20	1
34	3613272-00	Label, Adh Back, Mylar Cap	1
35	9007031-00	Tie, Cable Bundl. Dia 0-3/4"=101	36
36	9008264-00	Mount, Cable Tie, Adhesive Back	A/R
37	3621498-02	Label, Airflow CPU/NI CT 20	1
38	9105740-55	Wire (Wrap) 30AWG KYNAR UL14 (12 ft. required)	A/R
39	7428311-01	Support, Cable	1
40	9006659-00	Washer, Flat S/PAS	2
41	5415695-01	Current Limiter	1
42	7020488-00	Cable, Short Switch Vane	1
43	7021448-5C	Cable, DC Voltage Monitor Sect.1	1
44	3617674-00	Label, Serial/Power W/O UL + CSA	1
45	3617674-01	Label, Serial/Power W UL & CSA	1
46	3617880-09	Label, Class "A" Subassembly	1
47	3621501-02	Label, Module Location, NI20	1

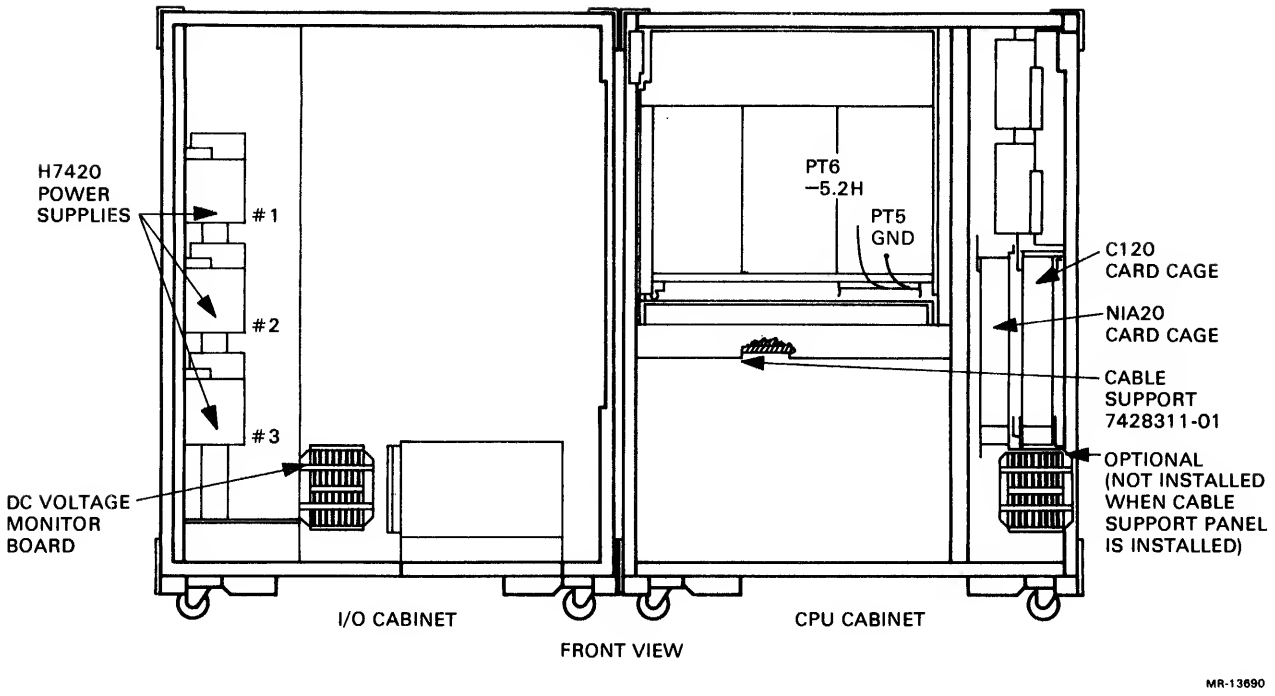


Figure 3-2 NIA20 in KL10-E, Front View

Table 3-2 NIA20 in KL10-E Harness and Cable Connections

Harness Connections

Parts List Item No.	Harness Terminals Point	Connection	Connection	Remarks
12	--	P1	NIA20 BP J2	--
	--	P3	NIA20 BP J1	--
	5	--	CPU #3 BP GND	--
	6	--	CPU #3 BP -5.2H	--
	--	P2	SECT. N-2 J1	Parts List Item 13
13	--	P1	H7440 J1	See Figure 3-5
	7	--	See Figure 3-5	
	8	--	See Figure 3-5	
30	--	P1	Fan Brkt. J1	See Note 1
	--	P2	See Figure 3-11	See Note 1
	--	J1	See Note 2	See Note 1
	--	P3	NIA20 BP J6	See Notes 1 and 3
29	--	P2	NIA20 BP J5	
	--	P1	J1	Parts List Item 31
	6	--	+5V Mon.Bd. J1-5	See Figure A-2

Table 3-2 NIA20 in KL10-E Harness and Cable Connections (Cont)

Harness Connections

Parts List Item No.	Harness Terminals Point	Connection	Connection	Remarks
41	P3	P1	CI20 Cable Vane Switch	See Note 4
	P2	--	NIA20 Fan Brkt. J1	See Note 4
	P1	--	CI20 Fan Brkt. J1	See Note 4
43	--	J1	P1 Parts List Item 29	--
	--	P1	Mon.Bd. J1	Parts List Item 31

NOTES:

1. Items not needed when CI kit is installed:

Item	Qty	Part No.	Description
5	8	9007786-00	Retainer, U-Nut 10-32X
6	8	9006073-01	Screw, Mach Pan Phil 10-
27	8	9007651-00	Washer, Lock Ext ST
28	8	9006664-00	Washer, Flat SST
3	1	7428312-01	Bracket, Interface
30	1	7019862-00	Harness, Vane Switch

2. J1 on parts list, item 30 (harness, vane switch) connects with existing connector P4 (see Figure 3-11).

3. Relocate cable from CI20 card cage at J6 connector and insert into NIA20 card cage J6 connector.

Table 3-2 NIA20 in KL10-E Harness and Cable Connections (Cont)

Parts List Item No.	Cable Connections From Unit	Location	Ref.Desig.	To Unit	Location	Ref.Desig.	Remarks
10	C.Cage Fan Brk	Gnd J2	Gnd P2	25 or 26	J1	P1	NIA20-CB
14	C.Cage	J4	P1	41		P2	
25/26	Item 10	P1	J1	861 PC		P1	Connect to or 11 any avail. switched outlet
1/2	C.Cage	Gnd	-	Cabrail	Hole 1	-	
15	NI20 BP	J3	PI Stripe Down	RH20 DTE CC	M3003 J2	P2 Stripe Up	
32	M3003	J1	P3	M3002	J1	P2	
	M3001	J1	P1	M3002	J1	P2	
38	RH20 BP	B10N1	-	RH20 BP	B13B1	-	
		B13B1	-		B19B1	-	
		C10L2	-		B13B2	-	
		B13B2	-		B19B2	-	
		C10K2	-		B13U1	-	
		B13U1	-		B19U1	-	
		C10T2	-		C13B1	-	
		C13B1	-		C19B1	-	
		C12H2	-		C13N1	-	
		C13N1	-		C19N1	-	
		C12L1	-		C13B2	-	
		C13B2	-		C19B2	-	
		C14H2	-		A15R2	-	
		C14F1	-		F15A1	-	
		A14J2	-		A15E1	-	
		C14P1	-		A15D2	-	
		C14K2	-		A15S2	-	
		B14J1	-		B15A1	-	
		C20H2	-		A21R2	-	
		C20F1	-		F21A1	-	
		A20J2	-		A21E1	-	
		C20P1	-		A21D2	-	
		C20K2	-		A21S2	-	
		B20J1	-		B21A1	-	

The NIA20 installation uses assigned slots in RH20 logic assembly positions 4 and 5, with RH20 positions 6 and 7 reserved for installation of a CI20 computer interconnect. A system containing an NIA20 is limited to a maximum of four RH20s. In the installation of an NIA20, a module blank assembly, Digital P.N. 7019266-00, is used to prevent plugging any other module into RH20 position 4 as described in subsection 3.4.3, instruction 7 (Figure 3-3).

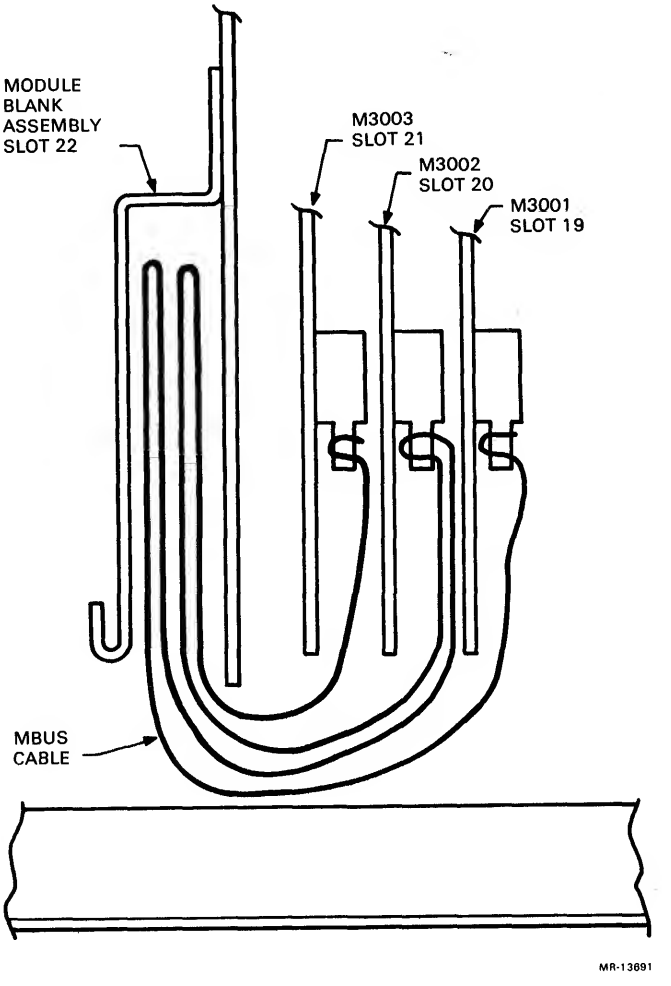


Figure 3-3 MBus Cable Interboard Connection, Top View

NOTE

Previous or subsequent installation of a CI20 with an NIA20 requires minor deviations from the procedure and will be described when applicable.

Installation of the NIA20 in an existing system requires implementing the following procedures, which are described in this chapter.

1. Unpacking and checkout of installation kit
2. Preinstallation checkout
3. Backplane wire adds
4. Installation of port modules
5. Installation of power supply regulator
6. Installation of NIA card cage
7. Installation of NIA current limiter
8. Installation of dc power harness
9. Installation of vane switch harness
10. Installation of dc voltage monitor harness and module
11. Installation of PLI bus
12. Installation of fan ac cable and power cord
13. Installation of internal NIA cable
14. Installation of KL10 adapter board and blank module assembly
15. Checkout.

3.2 UNPACKING AND CHECKOUT

Before unpacking any equipment, move all boxes into the computer area. Check the shipment against the packing list to be sure that all boxes were sent. If any boxes are missing, contact the customer and the branch field service manager. Check that all boxes are sealed, and there is no sign of external damage (dents, holes, or damaged corners).

If any boxes are open or damaged, document it on the installation or field service report and inform the customer. Open the boxes one at a time, starting with the box marked "READ ME FIRST" and find the packing slip. Check the contents of the box against the packing slip and examine each item for damage. Note missing or damaged items on the installation report or field service report.

At completion of the unpacking and checkout phase, advise the branch field service manager of any problems that occurred during this phase. If any items are damaged, the branch field service manager may want the customer to file an insurance claim. For missing items, the branch field service manager should get a short-ship request.

3.3 EQUIPMENT NEEDED FOR INSTALLATION AND CHECKOUT

The following equipment is required for installation and checkout of the NIA20:

1. Wire wrap tool (or wire wrap gun), No. 30 AWG, Digital P.N. 29-18301
2. Wire unwrapping tool, No. 30 AWG, Digital P.N. 29-13513
3. Regular Phillips screwdriver
4. Tektronix 475 oscilloscope or equivalent (100 MHz)
5. KLAD pack
6. Scope, digital voltmeter

3.4 INSTALLATION PROCEDURE

3.4.1 Preinstallation Checkout

Before performing the installation, verify that the configured system now in use is operating properly, to preclude the possibility of current system problems being ascribed to the NIA20 after its installation.

1. Remove all customer media, to minimize the possibility of corrupting customer data.

2. Mount the supplied KLAD pack; bring up the diagnostic monitor; and run the "B" string to verify that the system is working properly.

3. Power-down the system.

4. Verify that the system has a M8532-YA board installed. If not, replace the currently installed M8532 with a M8532-YA.

5. RH20 positions 4 and 5 will be used for the NIA20. If there is an RH20 in position 4, remove it. If there is an RH20 in position 5, leave it installed temporarily and perform diagnostic DFRHB to verify the reliability of the backplane wiring. If there is no RH20 in position 5, relocate a module from one of the other RH20 positions to position 5.

6. Power-up the system and run diagnostic DFRHB. This verifies that the backplane wiring of RH20 position 5 is functional. Power-down the system and reinstall the RH20 in its original position.

7. Perform diagnostic DFRHB also in RH20 position 7, to verify the reliability of existing backplane wiring in RH20 positions 6 and 7, before implementing any NIA20 modifications.

8. Power-down the system and reinstall the RH20 in its original position.

3.4.2 Backplane Wire Adds

For the installation of the NIA20, 24 new wires must be added to RH20 backplane positions 4 and 5. An examination of the RH20 backplane must be performed to confirm the physical addition of the wire wraps listed in Table 3-3. A check column is included in the table for the wire installer to record installation progress.

To prepare the wire adds, strip approximately 1 inch of insulation from the wire to allow sufficient turns to be made on the wire wrap post. After each wire is added, enter a check mark in the blank space adjacent to the wire listing in Table 3-3.

Table 3-3 NIA20 in KL10-E Wire Adds

Signal Name	From	To/From	To	Check
EBUS D11 L	B10N1	B13B1	B19B1	
EBUS D12 L	C10L2	B13B2	B19B2	—
EBUS D13 L	C10K2	B13U1	B19U1	—
EBUS PARITY L	C10T2	C13B1	C19B1	—
EBUS PI00 L	C12H2	C13N1	C19N1	—
EBUS PARITY ACTIVE L	C12L1	C13B2	C19B2	—
MPR7 MWBUSCTLFLD01 H	C14H2	A15R2		—
MPR7 MWMGCFLD08 H	C14F1	F15A1		—
MPR7 MWTIMEFLD H	A14J2	A15E1		—
CBI1 CLK2 L	C14P1	A15D2		—
CBI2 CLK4 L	C14K2	A15S2		—
CBI2 CCCHANERR L	B14J1	B15A1		—
MPR7 MWBUSCTLFLD01 H	C20H2	A21R2		—
MPR7 MWMGCFLD08 H	C20F1	F21A1		—
MPR7 MWTIMEFLD H	A20J2	A21E1		—
CBI1 CLK2 L	C20P1	A21D2		—
CBI2 CLK4 L	C20K2	A21S2		—
CBI2 CCCHANERR L	B20J1	B21A1		—

To assure the reliability of the new wiring, an ohmmeter check of each new wire add should be performed by a person other than the wire installer.

3.4.3 Installation of Port Modules

Protective backing is placed on the lower third non-component side of each port module and the upper third of the non-component side of the M3002. As the modules are inserted and/or removed, the protective backing protects the MBus and PLI bus cables. The protective backing should not interfere with the card guide or cover the gold finger contacts on the module. Insert the port modules as follows:

1. Connect MBus cable, Digital P.N. 7019270-1J. Be sure to orient the cable so that the flat wire comes out of the cable header away from the board, as shown in Figure 3-3.

2. Insert the M3001 EBus interface/port ALU module in the rightmost slot of RH20 position number 5 (slot 19, looking at the backplane from the module side). The arrow on cable should be aligned with the arrow on the board connector.

3. Connect the MBus cable to the M3002 Port microprocessor module as shown in Figure 3-3.

4. Insert Module M3002 in slot 20 to the left of the installed M3001 as shown in Figure 3-3.

5. Connect the MBus cable to the M3003 module as shown in Figure 3-3.

6. Install the M3003 CBus/PLI interface module in slot 21, which is located to the left of the installed M3002.

7. Install the module blank assembly, Digital P.N. 7019266-00, in RH20 position 4, slot 22. This assembly blocks slots 22, 23, and 24. It prevents modules from being inserted into RH20 position 4 and provides a baffle for system cabinet airflow.

8. Perform an ohmmeter check between PT17U and ground to verify that there are no shorts to ground.

9. Fold the MBus cable into the module blank assembly as shown in Figure 3-3.

10. Close the module door.

11. Attach the self-sticking module utilization decal, Digital P.N. 3622344-02, on the upper rear baffle panel.

12. Power-up the KL10.

13. Readjust the existing +5 V power supply to 5.0 $\pm$ 0.25 V. This adjustment is located on H7420 number 1 in H744 number 4. The location of this regulator is farthest away from the circuit breaker. The voltage is monitored at +5H, between PT17U and ground.

14. Type MR (CR) with KLDCP loaded and running, then type FX1 (CR) in response to the command prompt, as shown below:

    >. MR (CR)

    >. FX1 (CR)

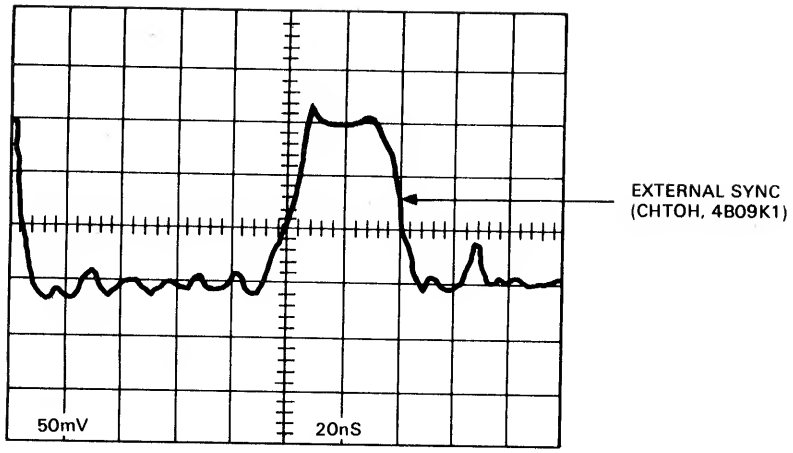
15. De-skew the port modules using a Tektronix 475 (or equivalent 100 MHz minimum) oscilloscope by performing the steps given in Figures 3-4 and 3-5.

16. Connect channel 1 of the oscilloscope to MTR MBOX CLK H, 4D33P1, on the CPU backplane. Use a ground clip.

17. Set the time base to 20 ns.

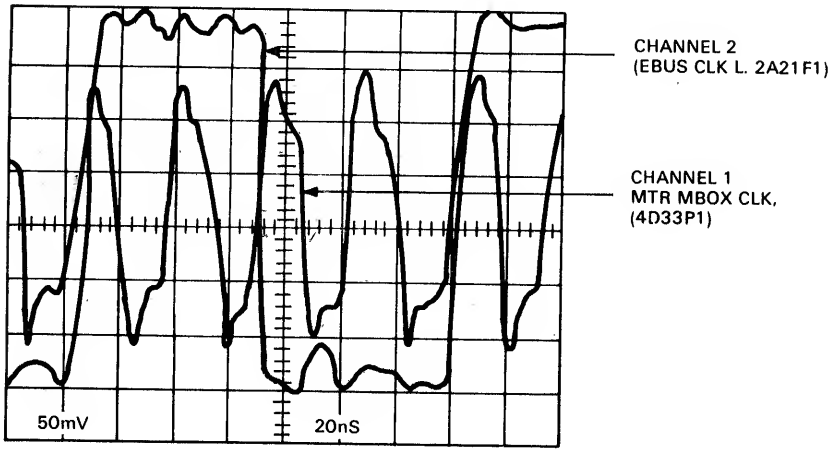
18. Set channel 1 vertical gain to 0.5 V/division. Set the ground reference to 1.3 V above the horizontal center level of the oscilloscope. (MTR MBOX CLK H is an ECL signal.)

19. Set the oscilloscope sync to positive external.



EXTERNAL SYNC (CHTO H)

Figure 3-4 NIA20 De-skew Timing. External Sync (CHTO H)



EBUS CKL L AND MTR MBOX CLK

MR-9846

Figure 3-5 NIA20 De-skew Timing. EBUS CLK L and MTR MBOX CLK

20. Connect external sync input to CHT0 H, 4B09K1 on the CPU backplane (Figure 3-4). Use a ground clip.

21. Connect channel 2 to CDS1, EBUS CLK L, 2A21F1 on the I/O backplane. Set the channel 2 vertical gain to 0.5 V/division. Use a ground clip. To measure TTL voltages, set the ground reference to 1.5 V below the horizontal center line of the oscilloscope.

22. Press the trigger view switch of the oscilloscope and display the external sync. Adjust the display, so that the rising edge of the external sync aligns with the vertical center line of the oscilloscope.

23. Display MBOX CLK H, channel 1. Identify the rising edge of MBOX CLK H that occurs prior to the vertical center line of the oscilloscope. Display channel 1 and channel 2.

24. Put the KL10-E in the override fault state. Remove the I/O rear door to access the I/O backplane.

25. Locate the bottom potentiometer on the clock module (M8559) in slot 12 of the I/O backplane. Using this potentiometer, adjust the falling edge of channel 2, EBUS CLK L so that it crosses the rising edge of MBOX CLK H. This crossing occurs on the horizontal center line of the oscilloscope.

3RD FROM BOTTOM

26. Disconnect all probes.

27. Mount the KLAD pack on the front end RP06.

28. Load and run diagnostic DFPTA to verify proper functioning of the port modules. If the modules fail, troubleshoot as directed by the diagnostic. If the modules are functioning properly, continue with the installation.

3.4.4 Power Supply Regulator Installation

Three H7420 power supplies are located on the I/O cabinet side wall as shown in Figure 3-1. The H7440 regulator to be added is installed in the upper H7420 power supply location. This additional +5 V regulator is required to support the NIA20 card cage and is installed as follows (see Figure 3-6):

1. Remove the spare slot filler panel from slot 5 of the H7420 number 1 power supply. Save all existing hardware.

2. Take the new H7440 regulator from the kit and install the H7440 in slot 5 of H7420 number 1, using two screws on top and one thumbscrew at the bottom. (Some systems may use H744 or H7440 regulators.)

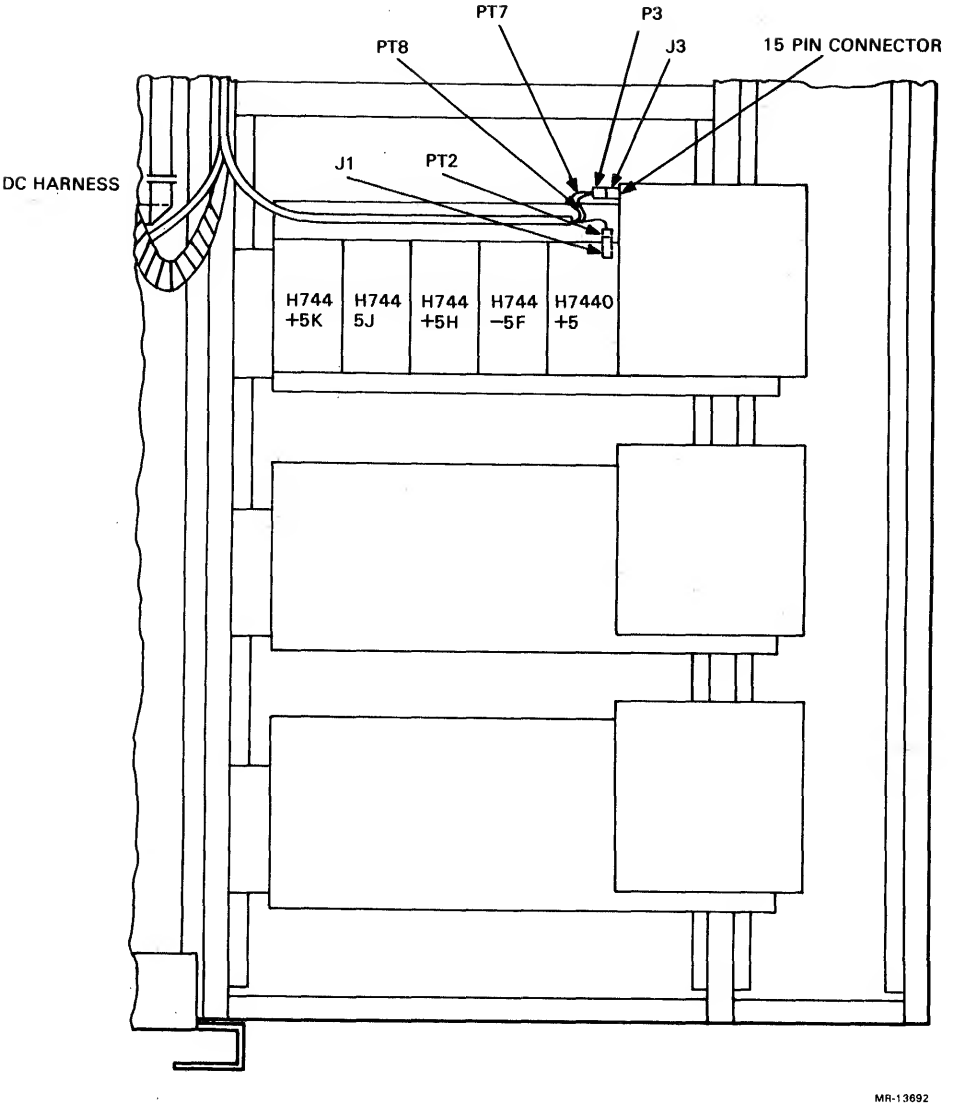


Figure 3-6 H7420 Power Supply

3.4.5 Installation of NIA20 Card Cage/Internal Cable

NOTES

1. When a CI20 is installed, the NIA20 is mounted as shown in Figure 3-1.

2. If the CPU cabinet side panel cannot be removed, the current limiter (see Section 3.4.5.1) should be installed before the card cage to avoid making the location inaccessible.

To install the NIA20 card cage shown in Figure 3-7 and the internal NIA20 cable:

1. Install the two NIA20 mounting brackets shown in Figure 3-8.

   a. Remove and reposition any tie-wrapped cables from the right-side frame member of the CPU cabinet (viewed from the front) to accommodate the NIA20 mounting brackets and card cage.

   b. Install a total of 8 U-nuts (Tinnerman nuts), Digital P.N. 9007786-00, on the right-side frame members of the CPU cabinet (viewed from the front) in preparation for NIA20 card cage and current limiter installation. Insert the Tinnerman nuts into frame holes 5, 11, 15, and 52 on each vertical side frame member counting up from the bottom of the cabinet (see Figure 3-8). Four Tinnerman nuts are inserted into each vertical side frame member.

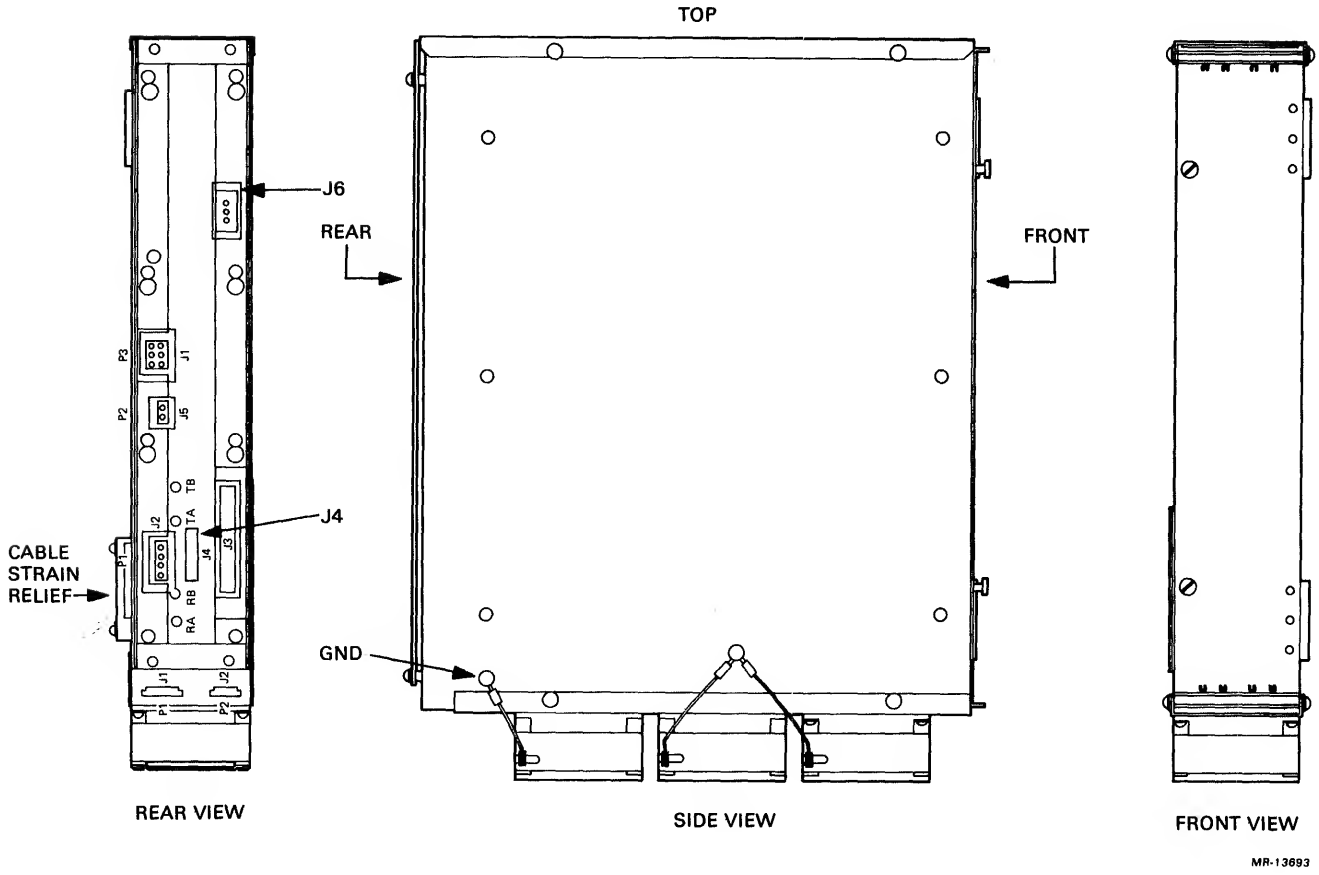


Figure 3-7 NIA20 Card Cage Views

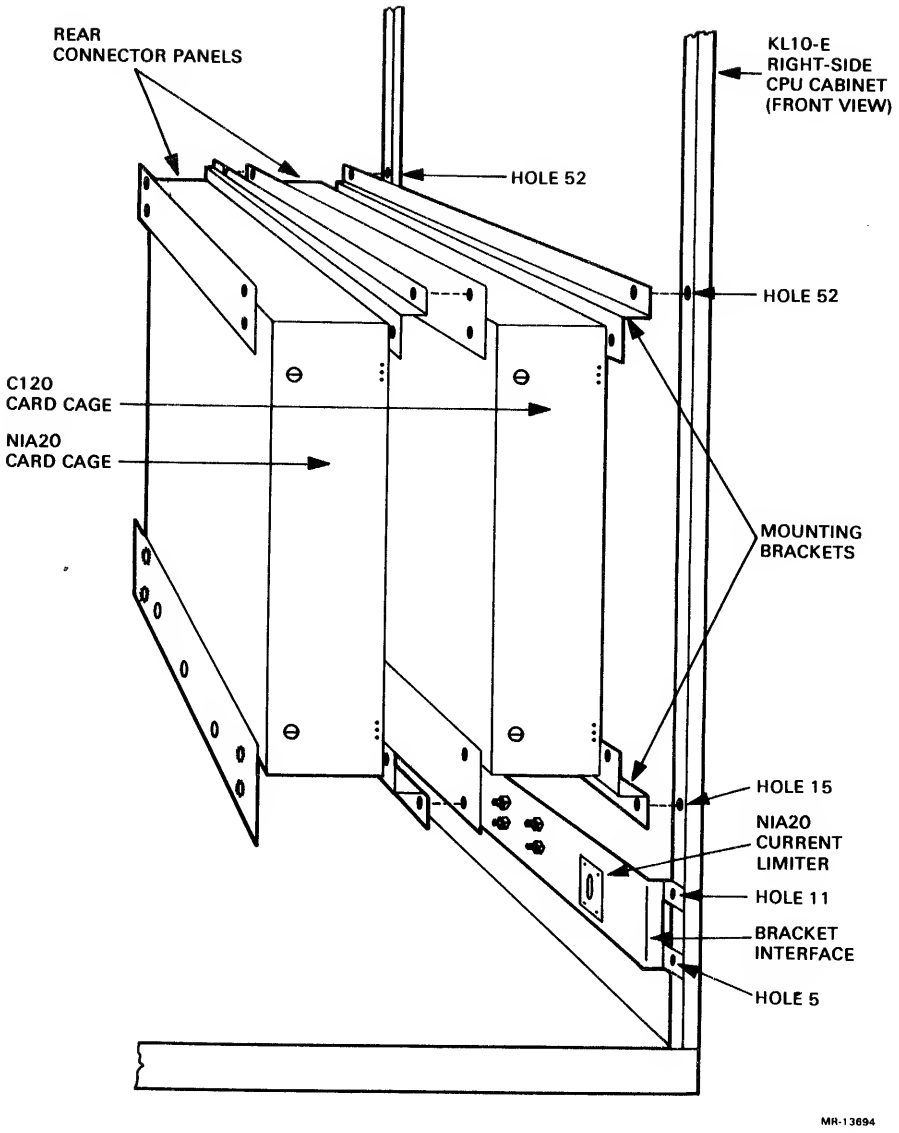


Figure 3-8 NIA20 Card Cage in KL10-E

c. Use four 10/32 one-half inch Phillips panhead machine screws, Digital P.N. 9006073-01 and four No. 10 star lockwashers, Digital P.N. 9007651-00 on each vertical side of the frame.

2. Locate the NIA20 internal cable strain relief (white) on the rear left-side of NIA20 card cage (see Figure 3-7). Because of the inaccessibility of this strain relief once the card cage is mounted, the internal cable is routed through it before installing the card cage.

3. Locate the three-foot internal NIA20 cable, Digital P.N. 7019893-2L.

4. Prepare the internal NIA20 cable for installation by positioning a stick mount on the right-side frame member of the CPU cabinet (viewed from the front), above the reserved CI20 connectors on the bracket interface (see Figure 3-9).

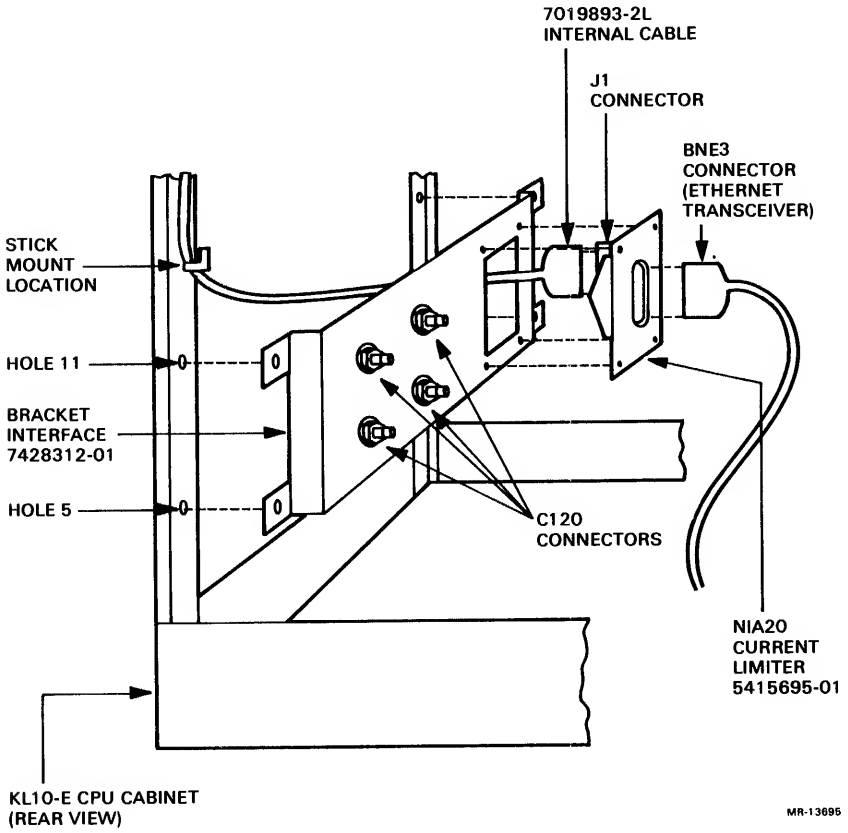


Figure 3-9 NIA20 Current Limiter

5. Route the three-foot internal NIA20 cable through the white plastic strain relief on the NIA20 card cage. Allow enough slack to connect the internal cable to the NIA20 card cage backplane and tighten the strain relief.

6. Connect the internal NIA20 cable to the J4 connector (see Figure 3-7) on the rear of the NIA20 card cage and route the cable as shown in Figure 3-1. The cable connector engages a detent when properly seated.

7. Mount the NIA20 card cage on the two NIA20 mounting brackets (see Figure 3-8), using a total of four 10/32 screws, external lockwashers, and flat washers in frame holes 15 and 52. Hang the NIA20 card cage on the top two screws; then install the bottom two screws.

8. Install the NIA20 card cage ground cable (see Figure 3-7).

   a. Install a Tinnerman nut in hole 11 of the left side frame member in the CPU cabinet (viewed from the rear).

   b. Connect the ground cable on the left-side frame member (viewed from the rear) by inserting a screw and using a starwasher on each side of the ground cable.

   c. Attach a ground label, Digital P.N. 3613272-00, closest to hole 1.

9. Run the internal NIA20 cable to the previously positioned stick mount and insert its other end into the rear J1 connector of the NIA20 current limiter.

3.4.5.1 **Installation of NIA20 Current Limiter** -- The NIA20 current limiter, Digital P.N. 5415695-01 is preinstalled on the bracket interface, Digital P.N. 7428312-01, as shown in Figure 3-9. The bracket interface, NIA20 internal cable, and BNE3 external cable are installed at the site as follows:

1. Locate the bracket interface, which is to be located on the lower right-side frame holes 5 and 11 of the CPU cabinet (viewed from the front). Four 10/32 screws, external lockwashers, and flat washers are used to install the bracket interface.

2. Connect the internal cable to the rear J1 connector and also connect the BNE3 external (Ethernet transceiver) cable to the front P1 connector located on the NIA20 current limiter (see Figure 3-9).

3.4.5.2 **Harness Installation** -- The following harnesses are to be installed:

1. Direct current power harness (two sections)
2. Vane switch cable
3. Direct current voltage monitor cables
4. Fan alternating current cable and power cord
5. PLI bus
6. External BNE3 NIA20 cable

The diagram in Figure 3-10 shows the harness and cable interconnections. The harnesses are installed as follows:

1. Install tie wraps approximately eight inches apart on all harnesses. When routing cables close to internal assemblies, use spiral wire-wrap to protect the cables from edges.

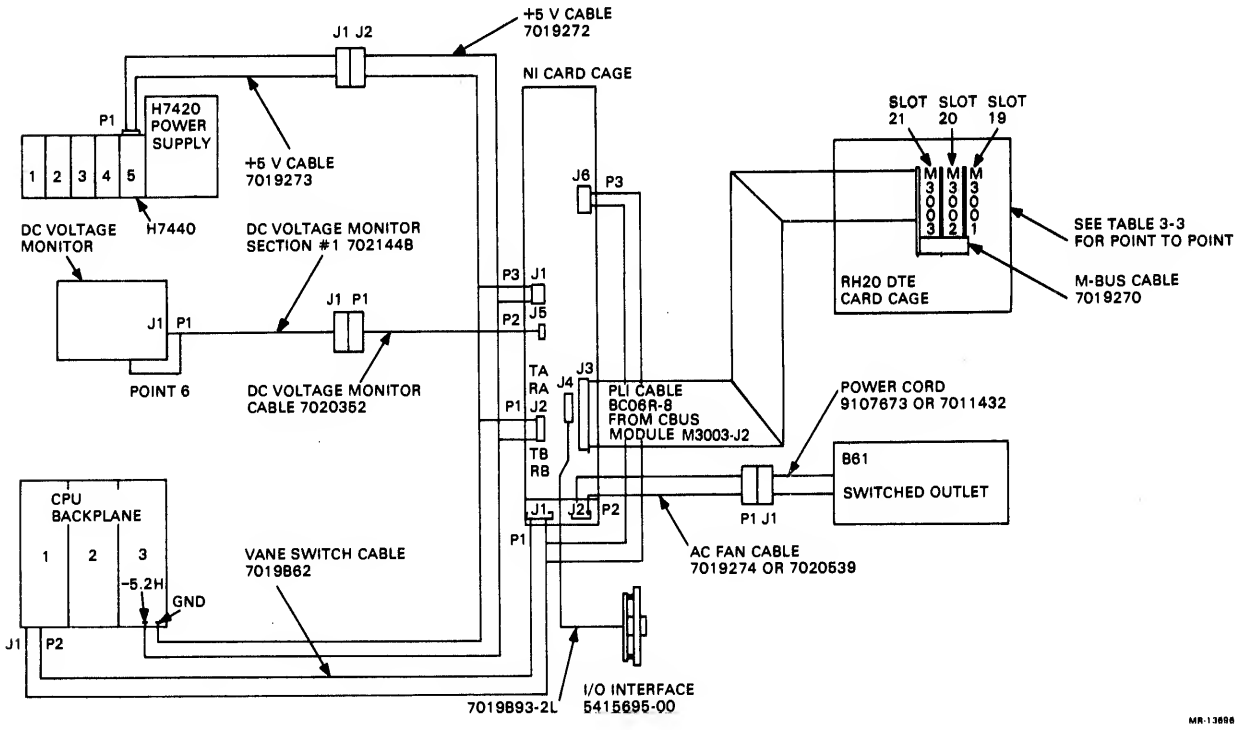


Figure 3-10 NIA20 Harness and Cable Interconnection Diagram

2. Locate the dc power harness, Digital P.N. 7019272-00 and 7019273-00 (see Figure 3-11), and its black and blue wires labeled PT5 and PT6. Connect the black wire to -5.2 ground and the blue wire to -5.2H in the CPU cabinet (see Figure 3-2).

3. Locate and connect P1 of the dc power cable, Digital P.N. 7019272-00, into connector J1 of the NIA20 card cage backplane (see Figure 3-7). Next, connect P3 of of dc power cable into J2 of the NIA20 card cage.

4. Connect P2 of the 7019272-00 dc power cable to J1 of the 7019273-00 dc power cable.

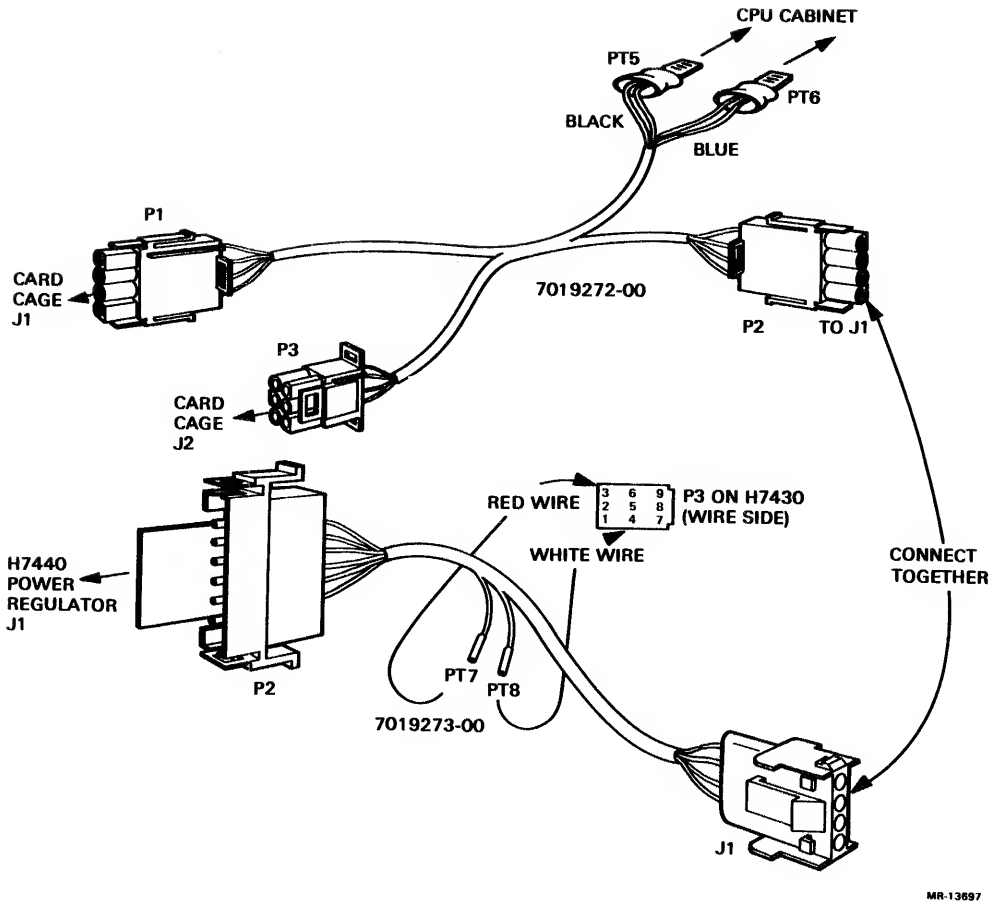


Figure 3-11 DC Power Cable

5. Tie-wrap the new harness to existing KL I/O power harnesses and route this cable as shown in Figure 3-1. Use spiral wrap along the harness where it contacts the side of the CPU frame member nearest the H7420 power supplies.

6. Locate the red and white wires labeled PT7 and PT8 of dc power cable (see Figure 3-11). Disconnect P3 atop power supply H7420 number 1, then connect PT7 and PT8 to pins 3 and 4, respectively, on P3 of the H7420. Then reconnect P3 to the H7420.

7. Connect P2 of the 7019273-00 dc power cable to connector J1 of the previously installed H7440 regulator (see Figure 3-6).

8. Locate the vane switch cable, Digital P.N. 7019862-00, (see Figure 3-12). Connect P1 of the vane switch cable to connector J1 located on the NIA20 card cage (see Figure 3-7). Also, connect P3 of the vane switch cable to connector J6 of the NIA20 card cage. Use stick mounts and spiral wire-wrap as needed to route and protect the vane switch cable.

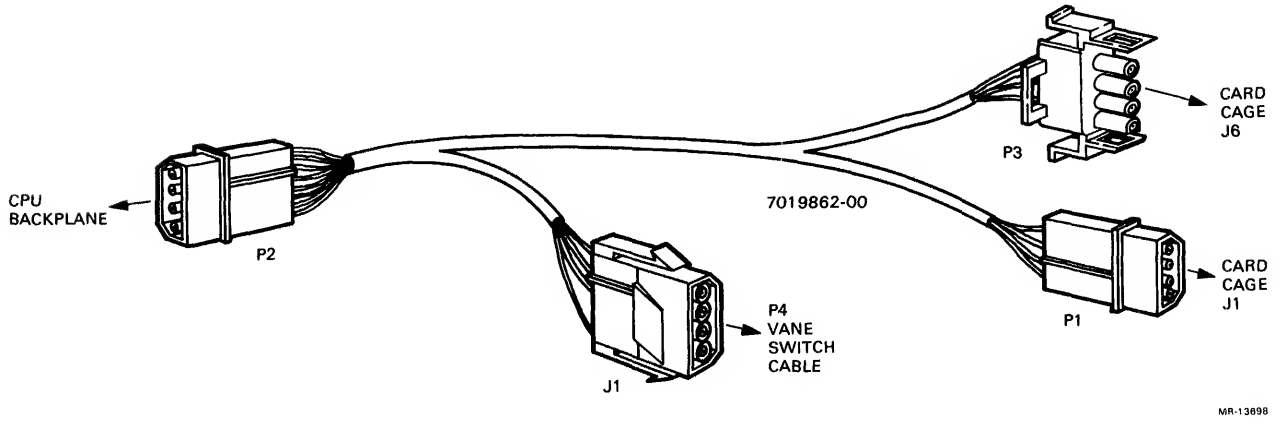


Figure 3-12 Vane Switch Cable

NOTE

When a CI20 is installed, the short switch vane cable, Digital P.N. 7020488-00, is used in a combined CI20/NIA20 installation. Consult the CI20 reference manual (Digital Order Number EK-CI20-RM-001) for other applicable CI20 installation procedures.

9. Remove the original KL CPU vane switch cable (P4) and connect this to the NIA20 vane switch cable connector J1.

10. Connect P2 of the vane switch cable to the original KL CPU vane switch assembly (see Figures 3-1 and 3-13).

11. Overlay the CPU/NIA20 air flow fault decal over the existing CPU air fault message decal on the 863 fault switch.

12. Locate the dc voltage monitor cable, Digital P.N. 7020352-00 (see Figure 3-14). Connect P2 of the dc voltage monitor cable to J5 on the NIA20 card cage and connect the other cable end (P1) into connector J1 on the new dc voltage monitor board.

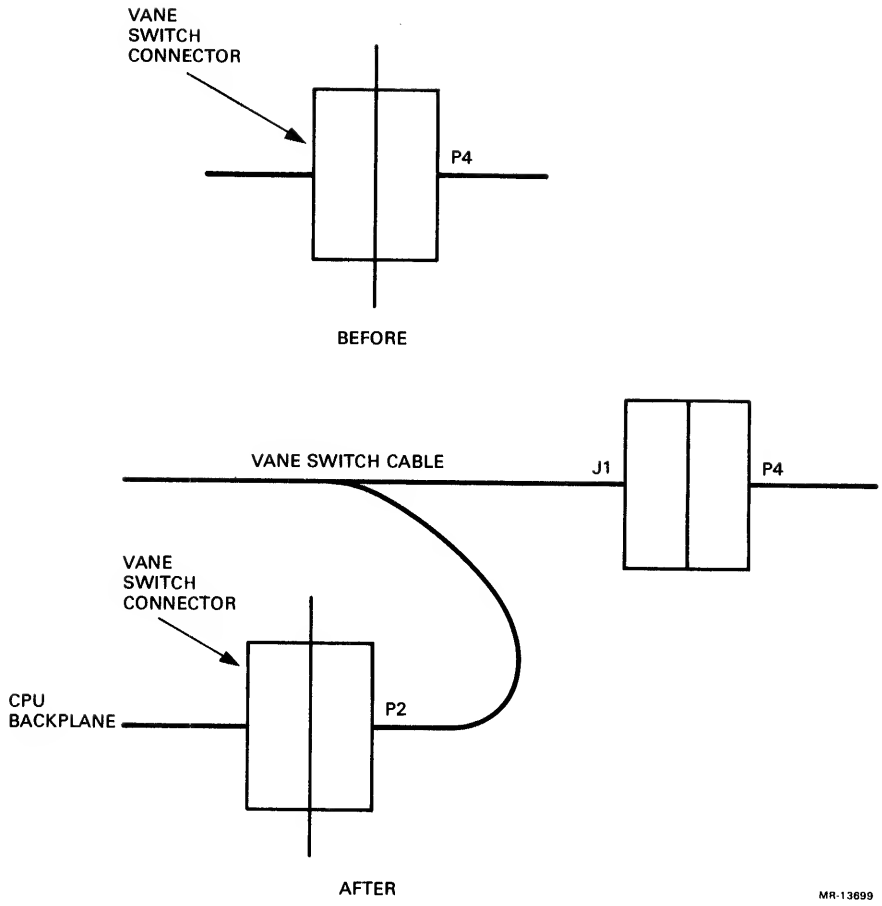


Figure 3-13 Vane Switch Harness Installation

13. Locate the switches on the dc voltage monitor board, Digital P.N. 5414506-01. Only switch S1 should be ON, while all other dc voltage monitor board switches should be OFF.

14. Insert the dc voltage monitor board into the +5 V slot of the dc voltage monitor card cage.

15. Attach the monitor panel decal, Digital P.N. 3621501-02, to indicate the slot used for the NIA20 dc voltage monitor board.

16. Connect the remaining single orange wire of the dc voltage monitor cable to a location adjacent to the existing orange wire on the dc voltage monitor board zone +5L.

17. Tie-wrap the dc voltage monitor and vane switch harnesses to the dc power cable. Use adhesive-backed square cable mounts to support the harness.

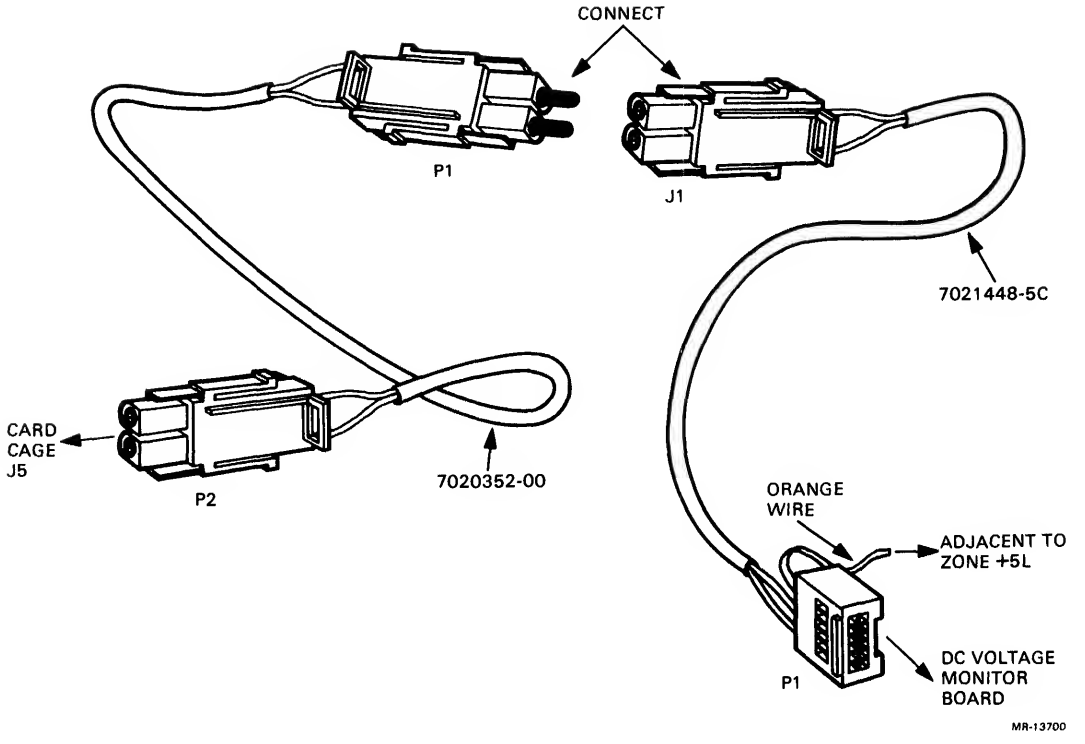


Figure 3-14 DC Voltage Monitor Cable

18. Locate the fan ac cable, Digital P.N. 7019274-06 (120 Vac 60 Hz) or 7020539-06 (240 Vac 50 Hz), and the power cord, Digital P.N. 9107673-06 (120 Vac 60 Hz) or 7011432-02 (240 Vac 50 Hz), (see Figure 3-15). Connect the fan ac cable connector P2 to connector J2 on the NIA20 card cage and then join the fan ac cable to the power cord. Insert the other end of the power cord into any available switched outlet of the 861 power controller. Connect the ground wire to the adjacent side ground screw on the NIA20 card cage. Use a starwasher to ensure a good electrical connection.

19. Install a Tinnerman nut in hole 11 on the frame and attach the ground cable from the NIA20 card cage to the frame. Use two starwashers to ensure a good electrical connection.

20. Locate the cable support panel, Digital P.N. 7428311-00 (see Figure 3-1). Install the smooth side of panel (used to support the cable harnesses) toward the rear when facing the CPU cabinet. Use four of each: Tinnerman nuts, screws, flat washers, and lockwashers in holes 36 and 39.

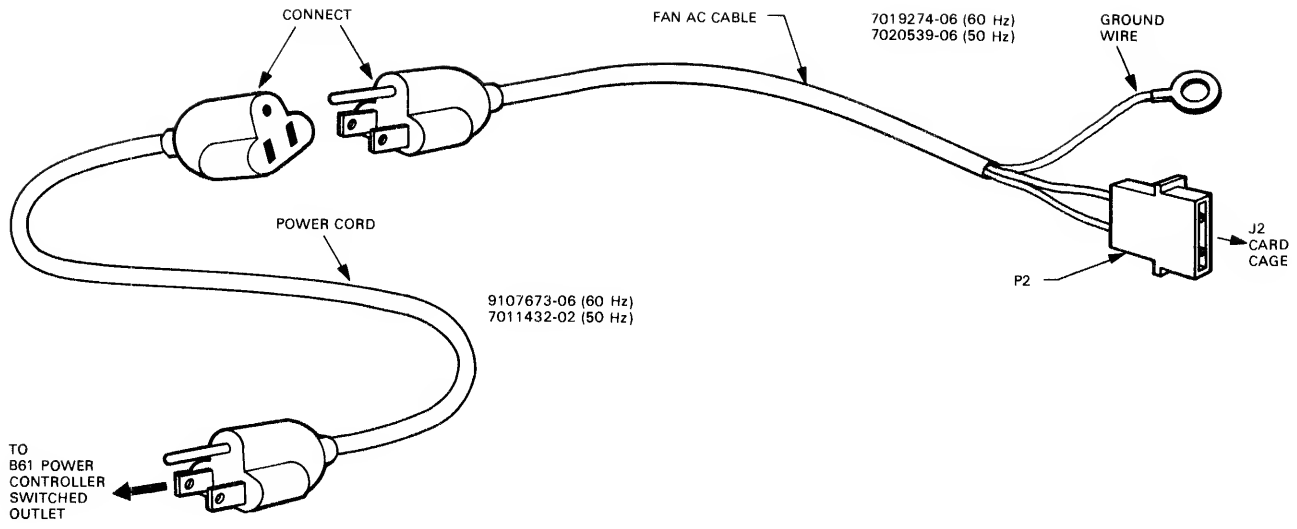


Figure 3-15 Fan AC Cable and Power Cord

21. Locate the PLI cable, Digital P.N. BC06R-08 (see Figure 3-1 for cable route and Figure 3-10 for cable connection). Connect one end of the PLI cable (identified by a red line imprinted on top of the cable) to module M3003 and route through the cable strain relief on the NIA20 card cage. The other end of the PLI cable (identified by a red line imprinted on bottom of the cable) to connector J3 on the NIA20 card cage (see Figure 3-7). To secure the PLI cable, install adhesive foam, Digital P.N. 1213716-00, within each of the four flat cable clamps. Install one cable clamp on the side of the CPU card cage and three cable clamps across the rear of cable support panel.

22. Route the cables as shown in Figure 3-1.

23. Replace the CPU cabinet door.

3.4.6 Installation of KL10 Adapter Board and Blank Module Assembly

The KL10 to NI adapter board, Digital P.N. L0072-00, and the blank module assembly, Digital P.N. 7014103-00, are installed in the NIA20 card cage as follows:

1. The KL10 to NI adapter board and the blank module assembly are installed into the NIA20 card cage by opening its front hinged-end panel door.

2. Install the KL10 to NI adapter board (L0072-00) in the LEFT t-hand slot.

RIGHT 3. Install the blank module assembly (7014103-00) in the adjacent slot to the [redacted].

3.4.7 Checkout

The physical part of the installation is complete at this point. All that remains is to verify that the system runs properly in the new configuration. Perform the following steps to verify the installation.

1. Verify that the KL10-E is no longer in the override fault state.

2. Power-up the KL10-E.

3. Readjust the 5 V power supply to 5.0 $\pm$ 0.25 V. This adjustment is located on power supply H7420 number 1, regulator H7440 slot 5 (see Figure 3-6). This regulator is located nearest the H7420 power supply breaker. The voltage is monitored at the black and red wires on connector J1 of the NIA20 card cage (see Figure 3-11).

4. Load and run diagnostic DFPTA for at least five passes in execute mode.

5. Load and run diagnostic DFNIE for at least five passes in execute mode.

6. Load and run diagnostic DFNIA for at least five passes in execute mode.

7. Enable the operating system.

8. Run diagnostics DFPTA for at least five passes in user mode.

9. Run diagnostics UETP NIA20 test for at least four hours in user mode.

10. Disable the operating system.

11. Remove all field service packs and tapes from the customer's system and store in a secure area.

12. Transfer and sign off system to customer's authorized representative.

CHAPTER 4
FUNCTIONAL DESCRIPTION

This chapter describes the port functions performed over the EBus, CBus, and PLI. It also provides a simplified NIA block diagram and sample transmit and receive operations.

Detailed descriptions of hardware components, register formats, microcode bit maps, and field descriptions, if not part of the descriptions in this chapter, are found in Chapter 5, Logic Description. Chapter 5 describes how the hardware and microcode implement the port functions.

4.1 PORT STATES

When a port reset is issued to the running port, the hardware is reset and the port microcode program counter is set to zero so that the power-up, self-check initialization code (see Chapter 5) can be executed. This guarantees that the port enters a well-defined state after the reset. All of the port registers are set to the power-up state. The port can be in one of the three states: uninitialized, disabled, or enabled.

4.1.1 Uninitialized

The port is not running -- the power-on state. The port enters this state after a power-up or a master reset. The port exits this state only after valid microcode is loaded into the port and the port clocks are started. Port clocks are enabled when the port driver sets the microprocessor run bit in the port control and status register (CSR32). If the port and microcode are functioning, the port driver sets the disable bit (CSR30), telling the port to put itself into the disabled state.

4.1.2 Disabled

The port is running but is not accepting NI packets. The port will process command queue entries that do not involve transmitting and receiving (that is, local commands). From the unitialized state, the disabled state is entered when the port driver sets microprocessor run and disable (CSR32 and CSR30). The port microcode then informs the port driver that it has put itself in the disabled state by setting disable complete (CSR12). From the enabled state, this state is entered by a command from the port driver to enter the disabled state, or when the port microcode detects a nonrecoverable internal port hardware error.

4.1.3 Enabled

The port is fully functional, processing commands and NI packets. This is the normal state for the port. This state is entered only from the disabled state, when the port driver sets the enable bit (CSR31). The port microcode informs the port driver that it has put itself into the enabled state by setting enable complete (CSR13).

4.2 CONTROL AND STATUS REGISTER

The KL10 and the port microprocessor exchange control and status information through the port CSR, which is a 36-bit register (see Figure 4-1). The CSR bits are defined in Table 4-1.

BIT NO.	BIT DEFINITION	RD/WR	
		KL10	PORT
00	PORT PRESENT	R	H
01	DIAG RQST CSR	R	H
02	DIAG CSR CHNG	R/H	H
03		*	*
04	RQST EXAM OR DEP	R/H	R/S
05	RQST INTERRUPT	R/H	R/S
06	CRAM PARITY ERR	R/C	H
07	MBUS ERROR	R	H
08		*	*
09		*	*
10		*	*
11	IDLE	R	R/W
12	DISABLE COMPLETE	R	R/W
13	ENABLE COMPLETE	R	R/W
14		*	*
15	PORT ID CODE 00	R	H
16	PORT ID CODE 01	R	H
17	PORT ID CODE 02	R	H

BIT NO.	BIT DEFINITION	RD/WR	
		KL10	PORT
18	CLEAR PORT	W	*
19	DIAG TEST EBUF	R/W	*
20	DIAG GEN EBUS PE	R/W	*
21	DIAG SEL LAR	R/W	*
22	DIAG SINGLE CYC	R/W	*
23	SPARE	R/W	*
24	EBUS PARITY ERR	H/R/C	R/H
25	FREE QUEUE ERR	R/C	R/S
26	DATA PATH ERR	R/C	R/S
27	CMD QUEUE AVAIL	R/S	R/C
28	RSP QUEUE AVAIL	R/C	R/S
29		*	*
30	DISABLE	R/S	R/C
31	ENABLE	R/S	R/C
32	MPROC RUN	R/W	R/H
33	PIA 00	R/W	R
34	PIA 01	R/W	R
35	PIA 02	R/W	R

* = NOT DEFINED
R = READABLE
W = WRITEABLE (SET OR CLEAR)
C = CLEARABLE ONLY
S = SETTABLE ONLY
H = HARDWARE CONTROLLED

MR-13775

Figure 4-1 Port Control and Status Register

The CSR is read/write interlocked to prevent the port and the KL10 from accessing it at the same time. When the port wants to access the CSR, it executes a request CSR microprocessor command. If the register is available, the interlock is asserted. If the CSR is not available because the KL10 is currently accessing the register with a CONI or a CONO, the interlock is not asserted until the CONI or CONO function is complete. The port microprocessor waits until the interlock is asserted before it attempts to access the CSR. In the same way, if the port microprocessor is accessing the CSR when the KL10 executes a CONI or CONO, the CONO/CONI waits until the port access is completed.

Table 4-1 Control and Status Register Bit Definitions

Bit	Name	Definition
00	PORT PRESENT	Indicates to the KL10 that the port is installed and powered-up.
01	DIAG RQST CSR	When set, this diagnostic bit indicates that the port has requested access to the CSR.
02	DIAG CSR CHNG	This diagnostic bit indicates that the contents of the CSR have changed since it was last read by the port microprocessor.
03	UNUSED	Not used by either the port microprocessor or the KL10.
04	RQST EXAM OR DEP	Used by the port microprocessor to request an EBus interrupt on PI level 00 (examine or deposit function). Setting this bit immediately generates the interrupt request.
05	RQST INTERRUPT	Used by the port microprocessor to request an EBus interrupt on PI levels 01 through 07. Setting this bit immediately generates the interrupt request.
06	CRAM PAR ERR	Indicates that a control RAM (CRAM) parity error has been detected. If this bit is set, the port microprocessor is immediately halted and RQST INTERRUPT (CSR05) is set. A hardware nonvectored (40 + 2n) interrupt will be forced. A CRAM PAR ERR may be forced in order to halt the port microprocessor at a specific location (break point). The port microprocessor cannot be restarted (CSR32 set) until this bit is cleared.
07	MBUS ERR	Indicates that more than one MBus driver has been turned on at the same time. That is, more than one set of port logic is trying to drive the MBus at the same time. If this bit is set the port microprocessor is immediately halted and RQST INTERRUPT (CSR05) is set. A hardware nonvectored (40 + 2n) interrupt will be forced. The port microprocessor cannot be restarted (CSR32 set) until this bit is cleared.

Table 4-1 Control and Status Register Bit Definitions (Cont)

Bit	Name	Definition
08	UNUSED	Not used by either the port microprocessor or the KL10.
09	UNUSED	Not used by either the port microprocessor or the KL10.
10	UNUSED	Not used by either the port microprocessor or the KL10.
11	IDLE	Indicates that the port microprocessor is in the idle loop, and is not hung in some other microcode routine.
12	DISABLE COMPLETE	Informs the KL10 that the port microprocessor has placed itself in the disabled state.
13	ENABLE COMPLETE	Informs the KL10 that the port microprocessor has placed itself in the enabled state.
14	UNUSED	Not used by either the port microprocessor or the KL10.
15 16 17	PORT ID CODE 00 PORT ID CODE 01 PORT ID CODE 02	Three-bit port identifier code field. Informs software that this is an NIA20 port and not an RH20 controller. Hard-wired so that: 00 = 0, 01 = 1, 02 = 1.
18	CLEAR PORT	When set by the KL10, this bit resets the port. The microprocessor is halted and all pertinent registers and control logic are placed in a reset state. The bit clears itself after the reset function is completed.
19	DIAG TEST EBUF	This diagnostic bit enables the KL10 to do an EBus interface loopback function by loading and reading the EBus buffer (EBUF). If the port is not running (CSR32 is reset) and this bit is set, a KL10: DATAO loads EBus data into the EBUF; DATAI places EBUF data on the EBus;
20	DIAG GEN EBUS PE	This diagnostic bit enables the KL10 to test the EBus parity checker by forcing it to decode an EBus parity error. When this

Table 4-1 Control and Status Register Bit Definitions (Cont)

Bit	Name	Definition
		bit is set, EBUS PAR ERR (CSR24) is also set on the same CONO, assuming there was no real EBus parity error.
21	DIAG SEL LAR	This diagnostic bit enables a KL10 DATAI to read the CRAM address contained in the latch address register (LAR). If this bit is set and bits 19 and 32 are reset, then the DATAI causes the LAR contents to be asserted on EBus D01-D12.
22	DIAG SINGLE CYC	This diagnostic bit enables the port microprocessor to be single cycled. If this bit is set and the KL10 sets MPROC RUN (CSR32), the port microprocessor executes one microcycle and halts. MPROC RUN will be cleared when the microprocessor halts. The current address to be executed is fetched from the RAM address register (RAR). The next address to be executed is stored in the LAR at the completion of the microcycle. The KL10 must read the address from the LAR and load it into the RAR before executing the next single cycle.
23	SPARE	Reserved for future software use.
24	EBUS PARITY ERR	When read by the KL10, this bit indicates that an EBus parity error has been detected. When written as a 1 by the KL10, this bit will clear itself and CRAM PARITY ERR (CSR06).
25	FREE QUEUE ERR	Used by the port to inform the port driver that there are no free queue entries available on the free queue.
26	DATA PATH ERR	Informs the port driver that the port microprocessor has detected an error in the direct memory access data path.
27	CMD QUEUE AVAIL	Used by the port driver to inform the port that it has placed a command queue entry on a previously empty command queue.
28	RESP QUEUE AVAIL	Used by the port to inform the port driver that it has placed an entry on the previously empty response queue.

Table 4-1 Control and Status Register Bit Definitions (Cont)

Bit	Name	Definition
29	UNUSED	Not used by either the port microprocessor or the KL10.
30	DISABLE	Used by the port driver to tell the port to place itself in the disabled state (set CSR12).
31	ENABLE	Used by the port driver to tell the port to place itself in the enabled state (set CSR13).
32	MPROC RUN	When set by the KL10, this bit causes the CRAM control register to reset and enables the port microprocessor clocks. The port starts cycling at the address contained in the RAR. The next and subsequent addresses will be fetched from the Am2910 sequencer.
33 34 35	PIA00 PIA01 PIA02	Three-bit KL10 EBus physical interrupt assignment (PIA) field (PI level 01 through 07).

4.3 EBUS

The port EBus control logic arbitrates the EBus protocol and the port microprocessor protocol for interfacing to the EBus, and performs synchronization between the two. Figure 4-2 shows EBus-to-port interface. Figure 4-3 shows the EBus signals and Table 4-2 describes the EBus signals.

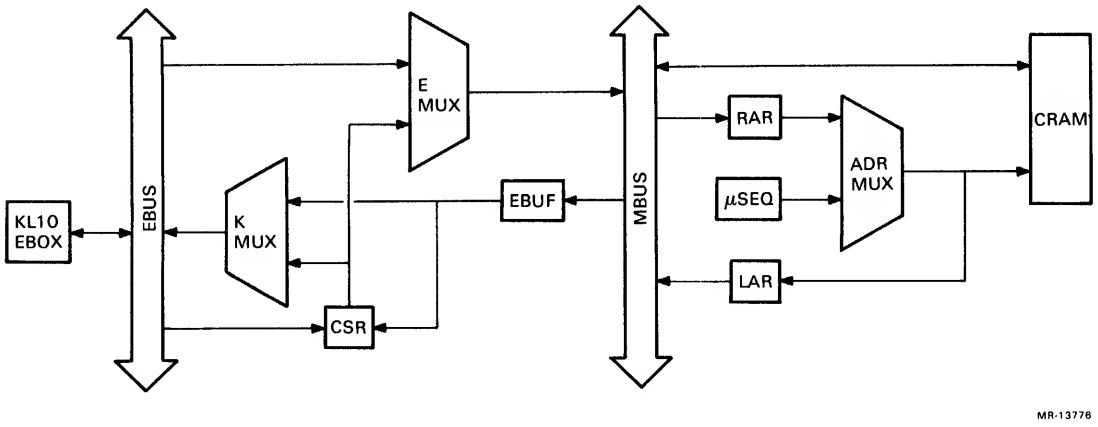


Figure 4-2 EBus-to-Port Simplified Block Diagram

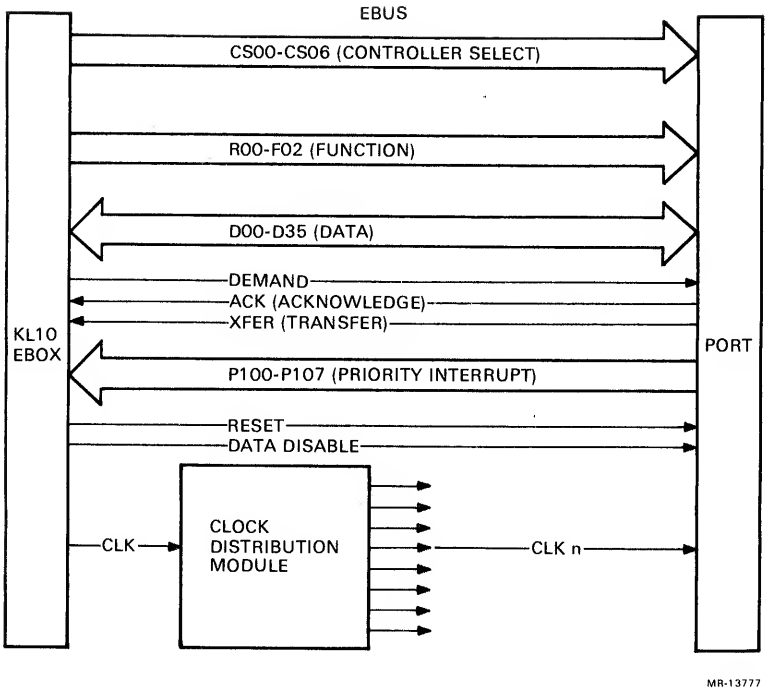


Figure 4-3 EBus Signals

Table 4-2 EBus Signal Description

Signal	Direction	Description
CS00-CS06	EBOX to port	Select the port by device code during a CONO, CONI, DATA0, or DATAI. CS04-CS06 specify the channel number during a PI SERVED and PI ADR IN. CS00-CS03 select the port by its physical number during a PI ADR IN.
F00-F02	EBOX to port	Specify the EBus command to be executed: **F00--F02 Command** 000 CONO 001 CONI 010 DATAO 011 DATAI 100 PI SERVED 101 PI ADR IN
D00-D35	Bidirectional data lines	Transfer control and status information between the EBOX and port.

Table 4-2 EBus Signal Description (Cont)

Signal	Direction	Description
DEMAND	EBOX to port	Causes the port to execute the command specified by F00-F02.
ACK	Port to EBOX	Indicates the port has received and is executing the EBus command; not asserted during PI SERVED.
XFER	Port to EBOX	Indicates the port has executed the EBus command; not asserted during PI SERVED.
PI00-PI07	Port to EBOX	Signals a port interrupt request by asserting the PIA (PI channel assignment) loaded in the port with a CONO.
DATA	EBOX to port	When asserted, disables EBus data line.
DISABLE		Disables drivers in the port to allow diagnostic operations to transfer diagnostic data.
RESET	EBOX to port	Initializes the port.
CLK	EBOX to port	Clock source for the port.

The KL10 has full control of the port only when the port microprocessor is not running (MPROC RUN, CSR32 is reset).

With the port in this state, the primary functions of the KL10 are to:

Load and read/verify the port microcode

Set up the correct initial CSR functions

Check for error conditions if the state of the port is due to an unexpected halt.

When the microprocessor is not running, the KL10 can also perform diagnostic functions, such as write and read/verify the E buffer, generate bad parity, and single-cycle the port. The KL10 performs these functions by executing CONOs, CONIs, DATAOs, and DATAIs, which are processed by the port. The specific functions are listed in Table 4-3. For a more detailed description of CRAM, RAR, and LAR, see Chapter 5.

Table 4-3 KL10 Diagnostic Functions

Function	Description
LOAD RAR	Load RAR. If the KL10 executes a DATAO with bit 00 = 1, the next port CRAM address is loaded from EBus D01-D13 into the port RAR.
LOAD MICROWORD	If the KL10 executes a DATAO with bit 00 = 0, the 28 least significant bits of the EBus are loaded into the selected half of the port CRAM location specified by the contents of the RAR.
READ MICROWORD	If the KL10 executes a DATAI and CSR21 = 0 (not DIAG SEL LAR), the contents of the selected half of the port CRAM location (specified by the contents of the RAR) are placed on the EBus.
READ LAR	Read LAR. If the KL10 executes a DATAI and CSR21 = 1 (DIAG SEL LAR), the CRAM address (contents of the LAR) is placed on EBus D01-D12.
LOAD EBUF	Load EBus Buffer. If the KL10 executes a DATAO and CSR19 = 1 (DIAG TEST EBUF), EBus D00-D35 are loaded into the EBUF.
READ EBUF	Read EBus Buffer. If the KL10 executes a DATAI and CSR19 = 1 (DIAG TEST EBUF), the contents of the EBUF to be placed on the EBus.
LOAD CSR	Load CSR. If the KL10 executes a CONO, the contents of the EBus are loaded into all of the CSR bits writable by the KL10.
READ CSR	Read CSR. If the KL10 executes a CONI, the contents of all CSR bits readable by the KL10 are placed on the EBus.

When the port microprocessor is running (MPROC RUN, CSR32 set), the KL10 can access the CSR only by executing CONO or CONI commands. With the port in this state, CONO and CONI commands (LOAD CSR and READ CSR) operate according to the description in Table 4-3.

The port ignores DATAO and DATAI instructions, executed by the KL10 software while the port is running (CSR32 set), as unexpected illegal functions. Therefore, an EBus timeout will occur, because the port does not return EBus Transfer (XFER). However, execution of a DATAO or DATAI by the KL10 microcode in response to a previous port examine or deposit request is not an illegal function.

4.3.1 EBus Interrupts

When the port is running, the port microprocessor controls the EBus by loading an IOP function control word (IOP word) into the EBus buffer and simultaneously generating an EBus interrupt. (The IOP word is identical to port error word 1 and equivalent to the API function control word.) (Figure 4-4 shows the format of the IOP word and Table 4-4 describes the bit functions.) The types of interrupts that may be generated by the IOP word are listed by function. The hardware can generate all of the interrupts, but the port microcode currently uses only bits 0, 4, 5, and 7.

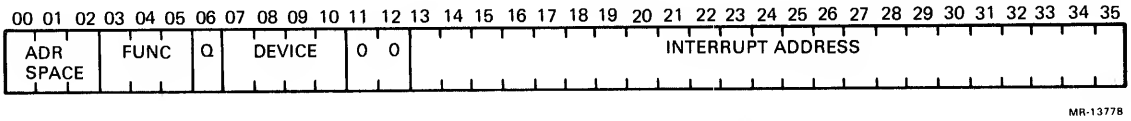


Figure 4-4 IOP Function Control Word

Table 4-4 IOP Function Control Word

Bits	Name	Description
00-02	ADDR SPACE	The address space containing the location addressed by bits 13-35, where: 0 = Executive process table (EPT) 1 = Executive virtual address space 4 = Physical memory All other codes are reserved.
03-05	FUNC	Function requested by the interrupt, where: 0 Standard (40 + 2n) interrupt 1 Not used 2 Not used 3 Not used 4 DATAO (examine) 5 DATAI (deposit) 6 Not used 7 DATAO (examine and increment). This formerly reserved function enables the NIA20 to manipulate queue interlocks.
06	Q	A qualifier is interpreted according to the function code, as follows:

Table 4-4 IOP Function Control Word (Cont)

Bits	Name	Description	
		Function	**Interpretation**
		0,7	Ignored
		4,5	Q = 1, apply protection and relocation to the address specified by bits 14-35
07-10	DEVICE	Physical device number assigned by the PI system.	
11-12	00		
12-35	INTERRUPT ADDRESS	The address at which interrupt handling begins.	

If the EBus interrupt is an examine or deposit request (function 4, 5, or 7), the port microprocessor requests the interrupt on PI level 00. PI level 00 examine/deposit interrupt requests are always processed by the KL10 (as highest priority), because PI level 00 interrupts cannot be selectively enabled or disabled as can interrupts on levels 01 through 07. Therefore, the KL10 processes port examine and deposit requests regardless of the enable or disable conditions of the KL10 PI system.

If the EBus interrupt is a function 0, the port microprocessor requests the interrupt on PI level 01 through 07 (as assigned in CSR bits 33-35).

Steps 1 through 9 outline the interrupt sequence.

1. If a previous interrupt is still active, the port waits before continuing.

2. The port then builds an IOP word and loads it into the EBUF. With the same microword, it requests either a standard interrupt or an examine or deposit interrupt.

   Examine or deposit interrupt (function 4, 5, or 7):

   The basic IOP words (less the interrupt address) for examine and deposit requests are predefined in the port local storage RAM. RQST EXAM OR DEP (CSR04) is set, causing the port to assert EBus PI request line PI00.

   Standard (40 + 2n) interrupt (function 0):

   The IOP word is all zeros for a standard interrupt request. RQST INTERRUPT (CSR05) is set, causing the port to assert the EBus PI request line (PI01-PI07) specified by the port PI level (PIA00-PIA02, CSR bits 33-35).

3. When the KL10 EBOX recognizes the PI request it responds by asserting

   The port channel number on EBus CS04-CS06

   PI SERVED (4) on EBus F00-F02

   EBus DEMAND, after a delay determined by the KL10.

4. The port responds by asserting the EBus data line corresponding to the port physical device number (EBus D07 for RH20 position 7, the CI20 position; or EBus D05 for RH20 position 5, the NI port position.)

5. After a KL10-determined delay, the KL10 EBOX reads the EBus data lines and negates EBus DEMAND.

6. Next, the KL10 EBOX asserts:

   The port channel number on EBus CS04-CS06

   The port physical device number on EBus CS00-CS03

   PI ADR IN (5) on EBus F00-F02

   EBus DEMAND, after a KL10-determined delay.

7. The port responds by asserting:

   EBus ACKN (acknowledge)

   The IOP word (previously loaded into EBUF by the port microprocessor) on EBus D00-D35

   EBus XFER, after a port-determined delay.

8. When the KL10 EBOX detects EBus XFER, it strobes the data from the EBus data lines and negates EBus DEMAND.

9. The trailing edge of EBus DEMAND causes the port to negate EBus ACKN, EBus XFER, and the EBus data lines.

4.3.2 Examine/Deposit Request Response

The KL10 microcode decodes the IOP word and executes the appropriate function. If the IOP word specifies a port examine or deposit request, then the first EBus cycle following the IOP word read will be a DATAO or DATAI to the port. The examine/deposit sequence is outlined in the following five steps:

1. The KL10 EBOX asserts

   A device code of zero (000) on EBus CS00-CS06

   DATAO or DATAI (2 or 3) on EBus F00-F02

Data on the EBus data lines, if a DATAO function

DEMAND, after a KL10-determined delay.

2. The port responds by asserting EBus ACKN, and asserting the EBus request condition code as a flag to the port microprocessor. (Condition codes are described in detail in Chapter 5.)

   The device code returned by the KL10 on EBus CS00-CS06 is zero, not the port device code, and is ignored by the port. The port does not examine the device code because it expects that this EBus cycle is either a DATAO or DATAI in response to its examine or deposit request. Therefore, as soon as the port senses EBus DEMAND, and when the port microprocessor detects the EBus request condition code, it does one of the following:

   In response to a DATAI, the port microprocessor places data on the EBus. (It must have previously loaded the data into the EBUF.)

   In response to a DATAO, the port microprocessor reads data from the EBus onto the MBus. On the same microcycle, it strobes the data from the MBus into internal storage.

3. After a port-determined delay, the port asserts EBus XFER.

4. When the KL10 EBOX detects EBus XFER it negates EBus DEMAND. If the function was deposit (DATAI), it strobes the data from the EBus data lines. If the function was examine (DATAO), it de-asserts the data from the EBus data lines.

5. The trailing edge of EBus DEMAND causes the port to negate EBus ACKN, EBus XFER, EBus request condition code, and the EBus data lines (if a DATAI function).

For examine/deposit functions, the port microcode responds to the EBus request condition code, in order to prevent EBus time-outs. The port microprocessor does not attempt to execute additional EBus transfers until it detects the negation of the condition code.

4.4 CBUS

The port CBus control logic arbitrates the CBus protocol and the port microprocessor protocol for starting and stopping the CBus, and performs synchronization between the CBus and the port mover/formatter. The CBus control logic also generates the clock timing for the port. Figure 4-5 shows to CBus-to-port interface,

Figure 4-6 shows the CBus, Table 4-5 describes the CBus signals, Figure 4-7 shows CBus timing, and Table 4-6 describes the CBus cycles.

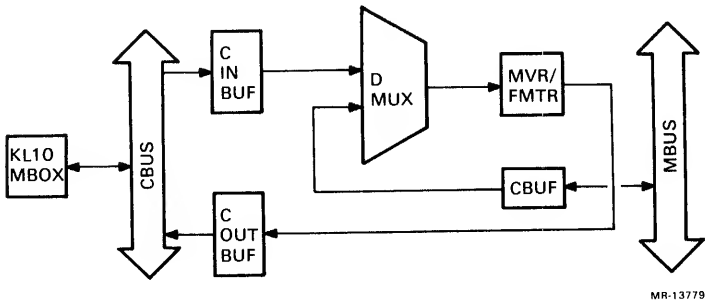


Figure 4-5 CBus-to-Port Simplified Block Diagram

The CBus (channel bus) is a synchronous, high-speed, time-division multiplexed, tristate data bus. It runs between the KL10 MBOX and the channel devices (the port, in this case -- see Figure 4-6 and Table 4-5). Each device on the CBus has a unique time slot. A CBus data transfer has four cycles: select, request, wait, and data (see Figure 4-7 and Table 4-6).

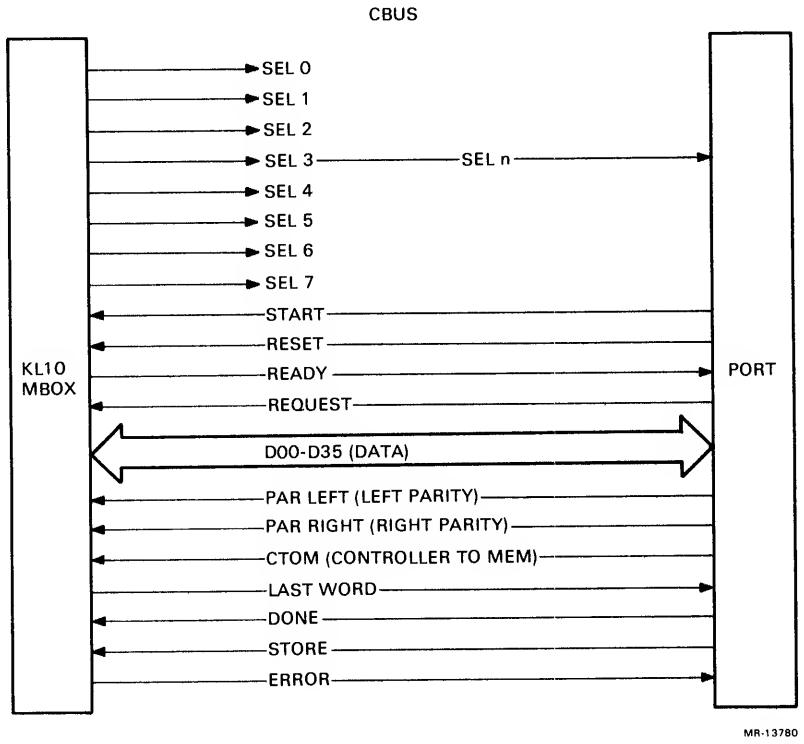


Figure 4-6 CBus Signals

Table 4-5 CBus Signals

Signal	Direction	Description
SEL 0-7	One line from the MBOX to each CBus device	Continuosly selects a different one of eight CBus devices. Defines the start of the four data transfer cycles (select, request, wait, data) for the selected device.
START	Port to MBOX	Asserted by the port (during its data cycle) to start a data transfer. The port asserts START during only one data cycle of the data transfer, and only if READY is negated.
RESET	Port to MBOX	Asserted by the port during the same data cycle that START is asserted. Causes the MBOX to: Clear the data buffers associated with the port Reset the command list pointer associated with the port to the initial channel control word Negate all the port status and data lines after the two previous steps are completed.
READY	MBOX to port	Asserted by the MBOX (during a data cycle) when it is ready for a data transfer, and after it detects START from the port. For an output data transfer, the MBOX has at least one data word in its buffer before asserting READY. READY is negated after the MBOX senses DONE and is prepared to start another data transfer. READY is also negated by errors.

Table 4-5 CBus Signals (Cont)

Signal	Direction	Description
		The asserted and negated states of READY apply only during data cycles. READY is said to be asserted when it is asserted during data cycles, and negated when it is not asserted during data cycles. The state of READY between data cycles is not pertinent.
REQUEST	Port to MBOX	The port asserts REQUEST during its request cycle if its CBus output buffer is full (device read -- data input to the KL10 and its CBus input buffer is empty (device write -- data output from the KL10). The port does not assert REQUEST if: The MBOX has not asserted READY The MBOX has asserted ERROR The MBOX has asserted LAST WORD during the current data transfer The port has asserted DONE during the current data transfer. For a device read, the port places data on the data lines (during its data cycle), and the MBOX strobes the lines at the trailing edge of the same data cycle. For a device write, the operation is reversed.
D00-D35	Bidirectional data lines	Valid only during a data cycle. If the port is not requesting data from the KL10, the MBOX places zeros on the data lines during the port's data cycle.

Table 4-5 CBus Signals (Cont)

Signal	Direction	Description
LAST WORD	MBOX to port	Asserted by the MBOX (during the data cycle) only for output data transfers at the same time the last data word is sent to the port. The port does not assert REQUEST after it detects LAST WORD.
DONE	Port to MBOX	Asserted by the port during its data cycle to terminate a data transfer. The port does not assert REQUEST after it asserts DONE.
STORE	Port to MBOX	Asserted by the port (during the data cycle, together with DONE) to store channel status in the channel's assigned reset and status logout area. STORE is asserted when the current transfer is terminated because of a port-detected error and the port microcode specifies that STORE be asserted.
ERROR	MBOX to port	Asserted by the MBOX (during the data cycle) to tell the port to terminate the current data transfer because of an MBOX detected error. When the port senses ERROR, it terminates the transfer by not asserting REQUEST, and asserting DONE in a subsequent data cycle.

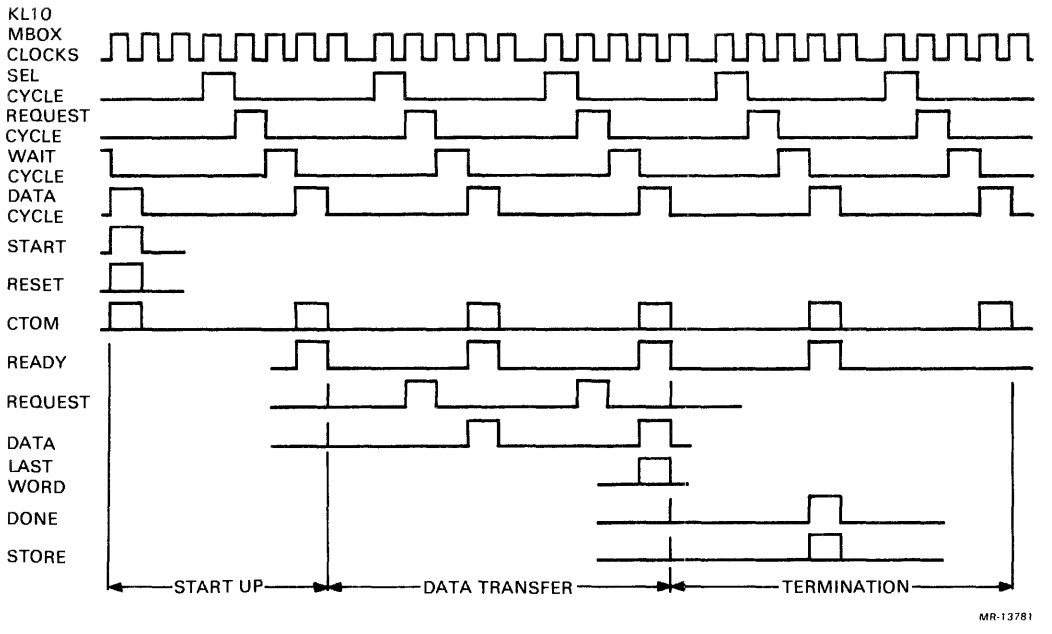


Figure 4-7 CBus Operation

Table 4-6 CBus Cycles

Cycle	Description
SELECT	There is a SEL line for each device on the CBus. The KL10 asserts CBus SELECT for the port when the port's time slot occurs. The port CBus control logic senses when its SEL line is active.
REQUEST	The port asserts CBus REQUEST during the entire request cycle time if the port has detected its SEL line active and is ready to make a data transfer. Otherwise, the port ignores this CBus data transfer.
WAIT	This is a dummy cycle for the port. It does not execute any CBus functions.
DATA	If data is to be transferred in this CBus data transfer slot, it is placed on the CBus data lines by either the KL10 (output data transfer) or the port (input data transfer) during the data cycle time. Otherwise, the port ignores the data cycle.

The port microcode prepares to start the CBus data channel by executing a start CBus microprocessor command. For an input data transfer to the KL10 memory, the port microcode also executes a write to KL10 memory microprocessor command. The CBus control logic latches these microprocessor commands until they can be

executed when the port CBus time slot becomes available. The CBus control logic detects the port time slot by sensing its CBus SEL line.

When the CBus control logic detects the port CBus SEL line asserted and the CBus READY line negated, it starts the channel by using the latched start CBus microprocessor command to assert CBus START and CBus RESET during the subsequent data cycle. The CBus control logic then clears the appropriate latches set by previous port microprocessor commands. If the transfer is to KL10 memory, the CBus control logic also uses the latched write to KL10 memory microprocessor command to assert CBus CTOM at this time. However, it does not clear the latch set by the write to KL10 memory microprocessor command until the data transfer is complete.

When the channel (i.e., KL10) is ready to transfer data over the CBus, it asserts CBus READY during the port data cycle.

After receiving CBus READY, the port CBus control asserts CBus REQUEST during its request cycle whenever it requires a data word from the channel (device write), or whenever it requires that the channel accept a data word (device read). The words are asserted on the CBus DATA lines during the port data cycle following its corresponding request cycle.

The port is ready to transfer data across the CBus whenever its CBus input buffer is empty, or whenever its CBus output buffer is full.

The CBus input buffer is emptied (transferred to the mover-formatter) with a CBus input buffer to formatter command microprocessor when the port microprocessor senses the CBus available condition code and is prepared to accept data from the CBus.

The CBus output buffer is loaded (the contents of the mover-formatter are transferred to it) with a formatter to CBus output buffer microprocessor command when the port microprocessor senses the CBus available condition code and has data available for transfer to the CBus.

When the channel places the last word on the CBus during a device write operation, it asserts CBus LAST WORD. In response, the port CBus control logic asserts the CBus last word condition code. When the port microprocessor detects this condition code, it responds with a stop CBus microprocessor command. This causes the port CBus control logic to assert CBus DONE during the next port data cycle. The port makes no more data requests during subsequent request cycles. CBus DONE causes the channel to terminate the operation. CBus READY is negated when the channel is prepared to begin another data transfer.

The port microprocessor can also execute a store CBus status information microprocessor command with a stop CBus command on the same microcycle. This causes the port CBus control logic to assert CBus STORE along with CBus DONE on the next port data cycle. Asserting CBus STORE and CBus DONE during the same data cycle forces the channel to store channel status in the channel's assigned reset and status logout area.

The port microprocessor executes both the stop CBus microprocessor command and store CBus microprocessor command, causing the port CBus control logic to assert CBus DONE and CBus STORE when it has transferred all data over the CBus during a device read operation. The port microprocessor also executes these commands during a read or write when it detects one of the following transfer error conditions (described in more detail in section 5.2):

CBus parity error
Mover/formatter parity check
CBus channel error
PLI parity error.

The port makes no more data requests during subsequent request cycles.

4.5 PLI

The port PLI control logic arbitrates the PLI protocol and the port microprocessor protocol for accessing the PLI, and performs synchronization functions between the PLI and the CMVR module. Figure 4-8 shows the PLI-to-port interface and Table 4-7 shows the PLI signals.

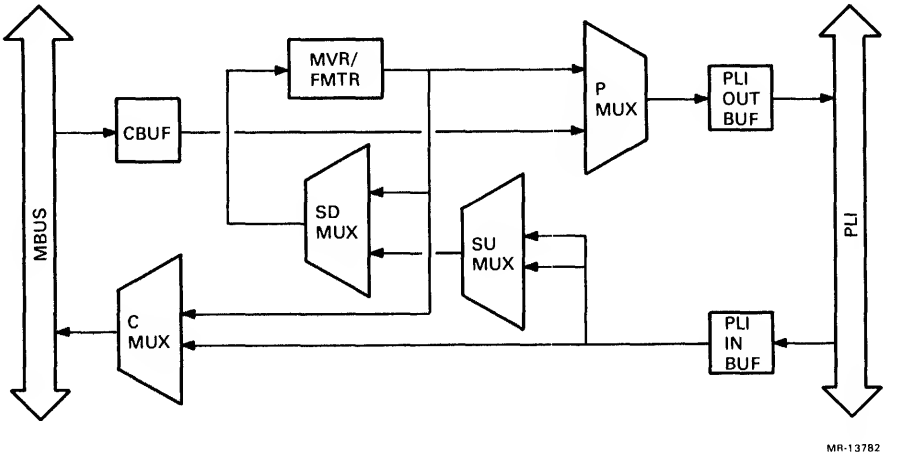


Figure 4-8 PLI-to-Port Simplified Block Diagram

Table 4-7 PLI Signals

Signal	Direction Port Link	Lines	Logic
DATA (7:0)	<-->	8	TRI-ST
SELECT	--->	1	TTL
RCVR ATTENTION	<---	1	TTL
END OF FRAME	<---	1	TTL
XMTR ATTENTION	<---	1	TTL
PLI/LINK CONTROL	--->	4	TTL
TRANSMIT PARITY	--->	1	TTL
RECEIVE DATA PARITY	<---	1	TTL
CLOCK	--->	1	TTL
INITIALIZE	--->	1	TTL
~~RECEIVER STATUS~~	~~<---~~	~~8~~	TTL
~~TRANSMIT STATUS~~	~~<---~~	~~8~~	TTL

The PLI interface provides the means for data transfers and communications between the port and the NIA to occur. The port has a master/slave relationship with the NIA. All data traffic and PLI functions are controlled by the port.

The following signal descriptions refer to some logic circuitry that has not been previously defined. (Chapter 5, Logic Description, contains additional information.)

4.5.1 PLI Interface Signals

The PLI interface signals are shown in Table 4-7.

4.5.2 Data (7:0) (Asserted High)

These lines are used to transfer data to and from the NIA and to pass control and status information to and from the NIA. The PLI/link control lines determine the direction and type of information being transferred.

4.5.3 Select (Asserted Low)

The select line must be asserted by the port to execute all data transfers and control functions. This line acts as an enable for the PLI/link control lines. The NIA provides a pullup resistor on this signal so that, if the port is not installed, the NIA does not respond to the floating control lines.

4.5.4 Receiver Attention (Asserted High)

RCVR ATTENTION is a PLI signal to the port. When asserted, it indicates that the receive status register contains a valid status on the next frame to be unloaded from the receive buffer. It also signifies that the frame buffer addresses are available on the used buffer list.

Receiver attention is asserted when the destination address of the frame is equal to the address stored in the physical address register or the multicast bit is set in the destination address of the frame. It is cleared by the reset receiver attention command (explained in Section 4.5.7.8).

4.5.5 End of Frame (Asserted High)

END OF FRAME is a signal to the port (data mover). When asserted, it indicates that the previous byte of data read from receive memory was the last byte of the frame. End-of-frame signal timing is the same as the data. The signal is asserted for one port clock cycle.

4.5.6 Transmitter Attention (Asserted High)

XMTR ATTENTION is a signal to the port processor. When asserted it indicates that the transmit status is available on the last frame transmitted and that the transmit buffer is available for the next frame to be loaded. Transmitter attention is cleared by the read transmit status command.

4.5.7 PLI/Link Control (Asserted High)

Four PLI/link control lines originating at the port are used to control the interface activities of the NIA. Control lines, denoted by asterisks, utilize the data lines to pass auxilary control information to the NIA. The SELECT line must be asserted to make the control lines valid. The control lines are encoded as shown in Table 4-8.

Table 4-8 Link Control Signals

Function	PLI/Link Control
WT XMIT BUF*	1100
XMIT ACTION*	0110
RD XMIT STATUS	1101
RD REC BUF	0010
RD REC STATUS	1110
RD USED BUF LST	1011
REC TO XMIT BUF	0011
RESET REC ATT	0111
ENABLE LINK CNTL*	1000
DISABLE LINK CNTL*	1001
WT REC BUF ADRS*	0101
WT FREE BUF LST*	0100
CLR RCV BUF	0001
WT ADRS*	1010
RD REG	0000
WT REG*	1111

* Control lines use the data lines to pass auxiliary control information to the NIA.

4.5.7.1 **Write Transmit Buffer (WT XMIT BUF)** -- The write transmit buffer function causes the data presented on the data lines and its associated parity bit to be written into the transmit buffer. The end-of-frame bit is always written as a zero. The buffer address counter will be incremented at the end of each cycle in which the load transmit buffer command is present. The load transmit buffer command is necessary for each byte transfer to the NIA.

4.5.7.2 **Transmit Action (FOUR COMMAND) Group** -- The transmit action command is a set of four commands whose action depends upon the state of port data bits 0 and 1. The four transmit action commands and corresponding data bit coding are shown in Table 4-9.

Table 4-9 Transmit Action Command Group

Command Name	Port Data Bit 1	0
XMIT FRAME	0	0
RESET TX BUF ADRS	0	1
TX BUF DEC	1	0
WT TX EOF	1	1

Command	Command Name	Function
XMIT FRAME	Transmit Frame	Informs NIA to begin transmission of frame stored in the transmit buffer; also clears the transmit status register.
RESET TX BUF ADRS	Reset Transmit Buffer Address	Resets the transmit buffer address counter to 0.
TX BUF DEC	Decrement Transmit Buffer Address	Causes the transmit buffer address counter to be decremented one count.
WT TX EOF	Write Transmit Buffer End-of-Frame Flag	Causes end-of-frame bit to be written as a one into the transmit buffer at the current address of the transmit buffer address counter.

4.5.7.3 Read Transmit Status (RD XMIT STATUS) -- This function enables the contents of the transmit status register onto the data lines. The transmit attention signal is cleared. (The transmit status register is described in Chapter 5.)

4.5.7.4 Read Receive Buffer (RD REC BUF)
This function enables the contents of the currently addressed location in the receive buffer onto the data lines. The read address counter is incremented at the end of each cycle in which this function is asserted. The parity bit for read data is passed to the port with the data on the receive data parity line.

The data is available from the receive buffer sequentially from the first byte received to the last byte received.

The port must, while reading the receive packet, monitor the signal END OF FRAME to determine when the last byte of the frame has been read.

4.5.7.5 Read Receiver Status Register (RD REC STATUS) -- This function enables the contents of the receive status register onto the data lines.

NOTE

The receive attention signal must be asserted in order to obtain a valid receive status. The contents of the receive status register are described in Chapter 5.

4.5.7.6 Read Receive Memory Used Buffer Address List (RD USED BUF LST) -- This command enables the first byte of the used buffer address list onto the data lines. The list contains addresses of data buffers used by the NIA during frame reception. They are provided to the port in the order that they were used by the NIA.

4.5.7.7 Transfer Byte from Receive Memory to the Transmit Buffer (REC TO XMIT BUF) -- This command causes the NIA to transfer one byte of data and its parity bit from the currently addressed location in receive memory to the currently addressed location in the transmit buffer. Both address counters are incremented at the end of each cycle when this command is executed.

4.5.7.8 Reset Receive Attention (RESET REC ATT) -- This command clears the receive attention signal. When this function is executed, the current receive status is lost. If there is another frame in the receive buffer, the status for that frame will be available when the receive attention signal is reasserted.

4.5.7.9 Enable Link Control/Disable Link Control -- These functions are used to enable and disable certain long-term functions in the NIA. A particular control may be set by executing ENABLE LINK CNTL with a 1 in the data line bit position

corresponding to that control. A control may be cleared by executing DISABLE LINK CNTL with a 1 in the proper bit position. Transfers with a 0 in any bit position have no effect.

4.5.7.10 Write Receive Memory Buffer Read Address to the Read Memory Address Register (WT REC BUF RD ADRS REGISTER) -- This command causes the data presented on the data lines to be written into the receive buffer read address register. The buffer address is combined with the read-receive-memory address counter to form a 14-bit address. All of the lower order bits are set to 0.

4.5.7.11 Write Free Buffer List (WT FREE BUF LST) -- This command causes the data presented on the data lines to be written into the free buffer list. It informs the NIA of free buffers in receive memory that are available (free) to store received data packets. The NIA uses the buffer addresses in the order they were received and combines them with the write-receive-memory address counter to form a 14-bit address.

4.5.7.12 Clear Receive Buffer (CLR RCV BUF) -- This command clears the entire free buffer list, used buffer list, and receive status first in, first out (FIFO). The command must be executed whenever a free buffer list parity error is detected. After the execution of this command, the free buffer list must be reloaded with buffer entries.

4.5.7.13 Write Address Register (WT ADRS REG) -- When this function is executed, the data lines must contain the address of the register or buffer to be accessed. The NIA will save the address. The transfer to or from the desired register will be executed when the RD REG or the WT REG command is given. Table 4-10 lists the registers and buffers available through the address register.

4.5.7.14 Read Register (RD REG) -- This function places the data of the register/buffer, whose address is stored in the address register, onto the data lines.

4.5.7.15 Write Register (WT REG) -- This function takes the data placed onto the data lines and writes it into the register/buffer whose address is stored in the address register.

4.5.8 Transmit Parity (Odd) (TTL Asserted High)
Odd parity is calculated by the port and transferred to the NIA using this line. The NIA stores the parity as supplied and checks parity when reading the buffer during a transmission.

4.5.9 Receive Data Parity (Odd) (TTL Asserted High)
Data being read from the receive buffer includes a parity bit that was generated before the data was written into the buffer. This parity bit is conveyed to the port via the receive data parity line and must be checked by the port.

Table 4-10 Write Address Register Access Table

ADRS (HEX)	Register Buffer	WT REG	RD REG
00*	PHY ADRS REG 0	X	--
01*	PHY ADRS REG 1	X	--
02*	PHY ADRS REG 2	X	--
03*	PHY ADRS REG 3	X	--
04*	PHY ADRS REG 4	X	--
05*	PHY ADRS REG 5	X	--
06 & 07	N/A	--	--
08	PHY ADRS ROM 0	--	X
09	PHY ADRS ROM 1	--	X
0A	PHY ADRS ROM 2	--	X
0B	PHY ADRS ROM 3	--	X
0C	PHY ADRS ROM 4	--	X
0D	PHY ADRS ROM 5	--	X
0E & 0F	N/A	--	--
10	XMIT BUF RD	--	X
11*	REC MEMORY WT	X	--
12	TDR REG LO	X	--
13	TDR REG HI	X	--
14	COLLISION TEST REG	--	X
15 THRU FF	NA	--	--

* The enable link bit in the link control register must equal 0 to access these addresses.

4.5.10 Clock Timing (PLI Bus)

The NIA interface requires a clock source from the port for its operation. The interface will operate with a minimum cycle period of 165 ns, as shown in Figure 4-9.

4.5.11 Initialize (TTL, Asserted High)

This signal from the port is used to initialize the NIA. A pullup resistor will be provided on this signal so that, if the port is not installed, the NIA will not interfere with the NI bus. The initialize signal must be asserted during power-up.

4.5.12 Receive and Transmit Status

These signals reflect the state of the receive and transmit status registers. They are brought out to backplane pins on the NIA link module. They are enabled with the receive and transmit attention signals. Chapter 5 contains a detailed description of the eight status bits.

4.6 SIMPLIFIED NIA BLOCK DESCRIPTION

This section uses the simplified NIA block diagram shown in Figure 4-10 to give an overview of the basic NIA functional operations: transmit a frame and receive a frame.

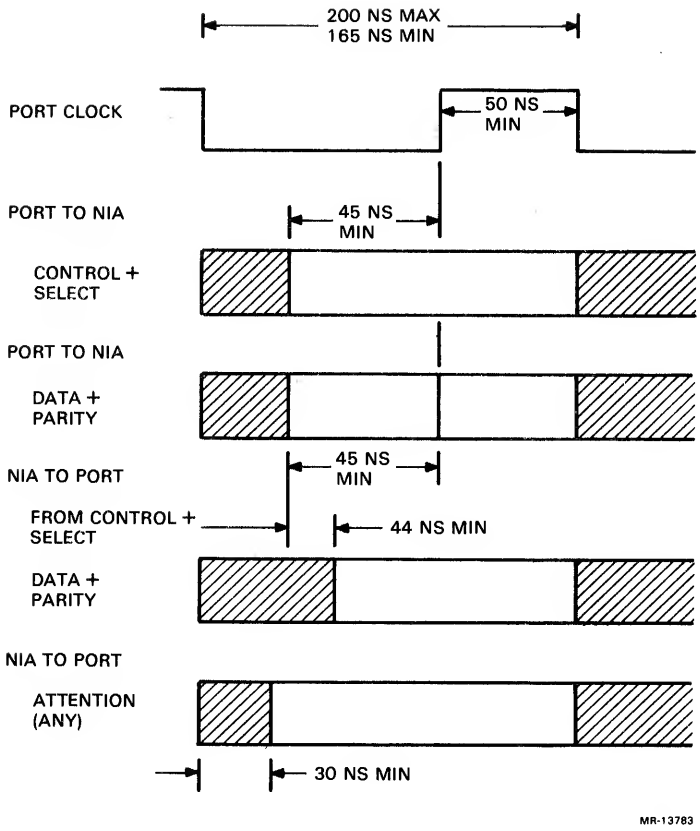


Figure 4-9 Clock Timing

The NIA module is functionally divided into a transmit section and a receive section, with a CRC function shared between the two. Not all logic blocks discussed in the following transmit and receive descriptions are shown on the simplified block diagram (status registers, counters, and address buffers). For this discussion, they are assumed. Chapter 5 presents the detailed NIA block diagram with a description of each subblock.

4.6.1 Simplified Transmit Operation

The port initiates a transmit function by setting the proper data and conditions in the NIA and issuing a transmit command. It then waits for the NIA to perform the transmission and assert Transmit Attention, which alerts the port to read the transmit status register and take appropriate action.

Initially, the port resets the transmit buffer address to zero, using the RESET TX BUF ADRS command. The port then loads the transmit buffer with 8-bit bytes and parity. The last word also sets bit 10 (bit 10 = load last byte = end-of-frame).

1. SELECT enables the link control lines.

2. The command WT XMIT BUF (via PLI/link control lines) causes the data on lines (7:0) to be written in the transmit buffer (plus parity).

3. Each write command automatically increments the address counter after writing the byte.

4. The command WRITE TX EOF writes the end-of-frame bit in the last location.

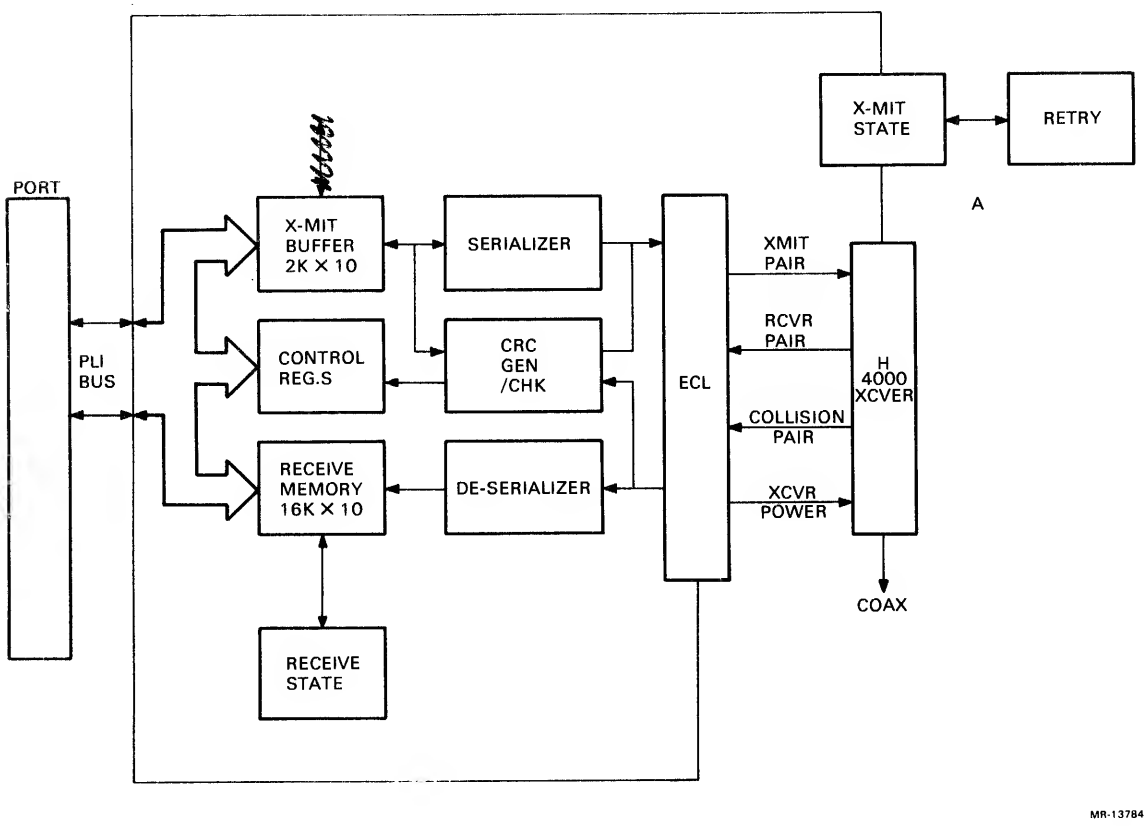


Figure 4-10 Simplified NIA Block Diagram

After the transmit buffer is loaded, the port initiates transmission by asserting the command XMIT ACTION on the PLI/link control lines, along with XMIT FRAME on the data lines. This command

1. Clears the transmit status register (last transmission status conditions)

2. Resets the transmit buffer address counter back to zero, (actually -8 to allow for the preamble, which is transmitted first)

3. Enables the NIA to transmit the frame from the transmit buffer.

The port now enters the idle loop and waits for the NIA to perform the transmission and assert XMIT ATT. The port can also receive and process other commands while waiting for XMIT ATT.

Upon receiving the command XMIT FRAME, the NIA checks to see if it is free to transmit. If carrier is present, meaning the wire is busy, the NIA defers and waits an additional 9.6 μs. When free to transmit, the NIA will start by reading the preamble ROM. The 8-bit ROM bytes are read out in parallel and serially shifted through the Manchester encoder (not shown, but in the emitter control logic section) and transmitted onto the wire.

Without allowing gaps in the serial bit stream, when the preamble is sent, the transmit buffer contents are likewise read, serially shifted, and transmitted. This data also passes through the CRC generator. In addition, parity is checked on the data out of the transmit buffer.

When the end-of-frame is detected (bit 10 set in the last byte read from the transmit buffer), the 32-bit CRC that was being generated is serially appended and transmitted.

During the transmission, the collision detect circuitry is monitered. If a collision was not detected and no other problems occurred, the transmit status bits will indicate a successful transmission on the first attempt. If a collision was detected, normal transmission is stopped, a short jam is transmitted, and the retry circuitry is enabled. Since the data is still in the buffer, it is available for retransmission (after a short, random delay). Again, the transmit status bits indicate to the port the action and conditions of the data transmission. After the frame has been transmitted, the NIA looks for the heartbeat signal from the transceiver. This signal indicates that the collision detect circuitry is working properly; if a collision had occurred, it would have been detected.

After the frame is transmitted, the NIA must complete the operation. During the transmission the NIA would have set bits in the transmit status register to specify the activities that occurred, either correct or faulty. The transmit status register contents can inform the port with the following terms:

DEFER -- Link had to defer to existing traffic on the wire before transmitting.

TRANSMIT RETRY STATUS -- The following transmit retry statuses are given:

1. Message transmitted on first attempt (no collisions).

2. Message transmitted on second attempt.

3. Message transmitted at some attempt after the second.

4. Message failed to transmit in 16 attempts, due to repeated collisions.

TRANSMIT ON TOO LONG -- Transmitter was on longer than it would take to transmit the longest valid frame, (asserted after 1536 bytes have been transmitted).

COLLISION INPUT FAILED -- Transceiver failed to assert the heartbeat signal after frame was transmitted.

TRANSMIT PARITY ERROR -- NIA detected a parity error while reading data from the transmit buffer (remainder of transmission aborted).

LATE COLLISION -- A collision occurred after the slot time of the channel has elapsed (no retries attempted and transmission aborted).

LOSS OF CARRIER -- Carrier not present on the channel throughout the transmission or the carrier dropped during transmission (the remainder of retrys are aborted).

The NIA completes its operations by asserting XMIT ATT to the port. Upon receiving XMIT ATT, the port exits from the idle loop, reads the transmit status register, and takes the appropriate action as indicated by the conditions given in the status bits. Normally, the port builds a response and puts it on the response queue to the port driver. Figure 4-11 shows the basic functions of a transmit operation summarized in a sequence flow diagram.

4.6.2 Simplified Receive Operation

At startup, when the port initializes the NIA, it sets initial conditions and returns to the idle loop. The NIA functions independently, receiving incoming frames and notifying the port by asserting RECEIVE ATTENTION. Initialization steps performed by the port include:

1. Clear the used buffer list (FIFO)

2. Load the free buffer list (FIFO) with addresses to be used during NIA frame reception

3. Set up the receive memory organization (usually 32 bit by 512 byte buffers)

4. Enable the NIA to receive frames (via link control register).

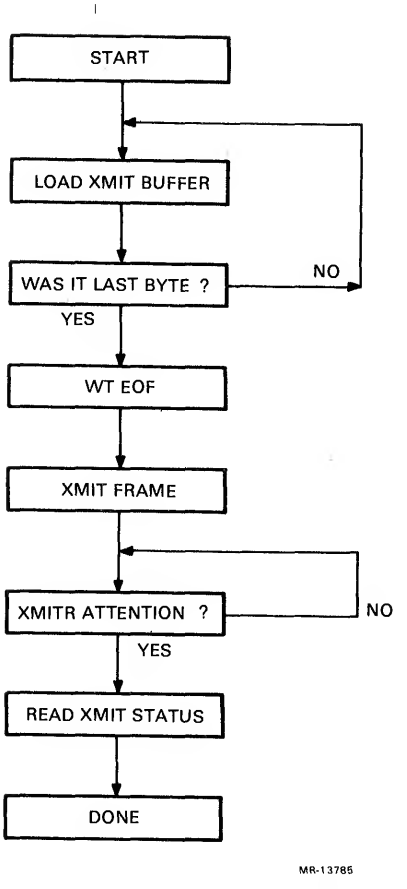


Figure 4-11 PLI Interface Transmit Flow Diagram

The NIA receive state machine rests in the idle mode, waiting to detect a carrier (activity) on the wire. When a carrier is detected, the receiver synchronizes on the incoming preamble and looks for the start signal, which is two consecutive ones.

The receive shift deserializer accepts the Manchester decoded serial bit stream and parallel-loads the data bytes (including a calculated parity bit) into the receive memory buffer. It loads the memory at the buffer address supplied from the next free buffer list of addresses. It also compares the incoming destination to the internal address, in search of a match. The serial bit stream is also sent to the CRC generator/checker.

If no match occurs, and it is not a broadcast or multicast address, the frame is not intended for this node. The NIA will discontinue writing into receive memory, reset the address counter to the beginning of the buffer, wait for no-carrier, and return to the idle state.

If a match condition had occurred, or the destination address was a broadcast or multicast address, the NIA must load the entire frame into the receive memory buffer. Each byte loaded will automatically increment the write counter, which contains the 8 low-order bits of the 14 memory addressing bits.

When a buffer boundary is encountered (i.e., 512 bytes have been received), the current buffer address (six high-order addressing bits) is transferred from the free buffer list to the used buffer list. The next address from the free buffer list (FIFO) is used to continue writing received bytes into memory.

When the last incoming bit is received, the carrier sense circuit detects the end of signal activity on the wire. This initiates the following events:

1. The NIA writes the end of frame, bit 10, in the next receive buffer location.

2. The CRC checker indicates CRC OK, (or CRC ERROR if bad CRC was calculated).

3. The 8-bit receive status is written to the receive status register (FIFO).

4. The current free buffer list address is removed from the free buffer list and put in the used buffer list.

5. The NIA receiver is set to the idle mode to await the next incoming frame.

6. The signal RECEIVE ATTENTION is asserted and sent to the Port.

When the port receives RECEIVE ATTENTION it exits the idle loop and processes the frame. The first step is to read the contents of the receive status register, which contains the receive conditions as seen by the NIA during frame reception. The eight status bits can indicate the following conditions:

1. PACKET FRAMING ERROR -- The frame did not contain a multiple of 8 bits; the CRC value at the last byte boundary was in error.

2. FRAME OVERFLOW ERROR -- The frame received was longer than the longest valid frame (over 1536 bytes).

3. CRC ERROR -- The circuit calculated bad CRC on the incoming frame.

4. FREE BUFFER LIST EMPTY -- The NIA could not store (or completely store) the incoming frame because no buffers were available from the free buffer list.

5. BUFFER USED BITS 0,1, and 2 -- The number of receive buffers used to store the received frame (one to six, depending on the frame size and memory organization being used -- 64/256, 32/512, or 16/1024).

6. FREE BUFFER LIST PARITY ERROR -- The link has stopped receiving frames due to corrupt data from the free buffer list.

If an error exists, the port takes appropriate action. It may decide to read (unload) the frame, or simply discard it (for example, runt frames are usually discarded). With no errors, the port starts unloading the receive memory buffer. Notice that the port now knows how many buffers were used to store the frame, by reading the used buffer bits.

The port reads the used buffer list addresses to learn where the frame is stored. It will then send the NIA the first memory address location to be read by writing the address into the receive memory read address register. The port will then execute a read receive buffer command to the NIA. Within the NIA, the receive memory buffer contents will be put on the port data lines (7:0), parity on the separate parity line; the address counter is automatically postincremented.

The port continues in this manner until 512 bytes have been read out or end-of-frame is detected. After 512 bytes, the port must write the next buffer address pointer to the receive memory read address register before reading more received data bytes.

This process continues (another 512 bytes read, update pointer, and so on) until the end of frame bit is detected. The end-of-frame bit signals the port that the previous read command produced the last byte of the frame. Since the port has the desired frame data, it now writes the used buffer list addresses to the free buffer list and resets receiver attention.

The port/NIA receive operation is now complete, but the port must continue the receive operation by building a response to pass the packet to the port driver. The NIA receive memory is time multiplexed between the NIA and port. During receive operations, the NIA can write a word and the port can read a word during one port clock cycle. These simultaneous operations are performed at different memory addresses and are independent. Also, in addition to the error and status detection circuitry, the following diagnostic and reliability features are employed:

1. The port can transmit a loopback frame. The data is transmitted through the Manchester encoder and looped back through the Manchester decoder and into the receive memory buffer. The data is then read from the buffer by the port.

2. The port can disable the CRC, which is used in the loopback mode (where the CRC circuit is dedicated to the receiver).

3. The port can transmit with its own address as the destination (or include itself in a group address). The data is sent on the wire and received from the wire as in normal operation. (In this case, the port must make allowances for the single CRC generator/checker, as explained in Chapter 5.)

4. The port can write data directly into the transmit buffer or receive memory buffers and then read the buffers back into the port.

5. The port can set promiscuous mode, which causes the receiver to receive all frames, regardless of destination address.

6. The port can set generate wrong parity on the data to be written into the receive memory buffer to allow a check of the parity check circuitry.

7. The port can read the time domain reflectometry register, which is useful in locating defective sections of cable.

8. The port can enable or disable the heartbeat signal.

9. The port can enable the collision test register (only in internal loopback mode) to force collisions and immediate retries (next slot time).

Figure 4-12 shows the basic functions of a receive operation summarized in a sequence flow diagram.

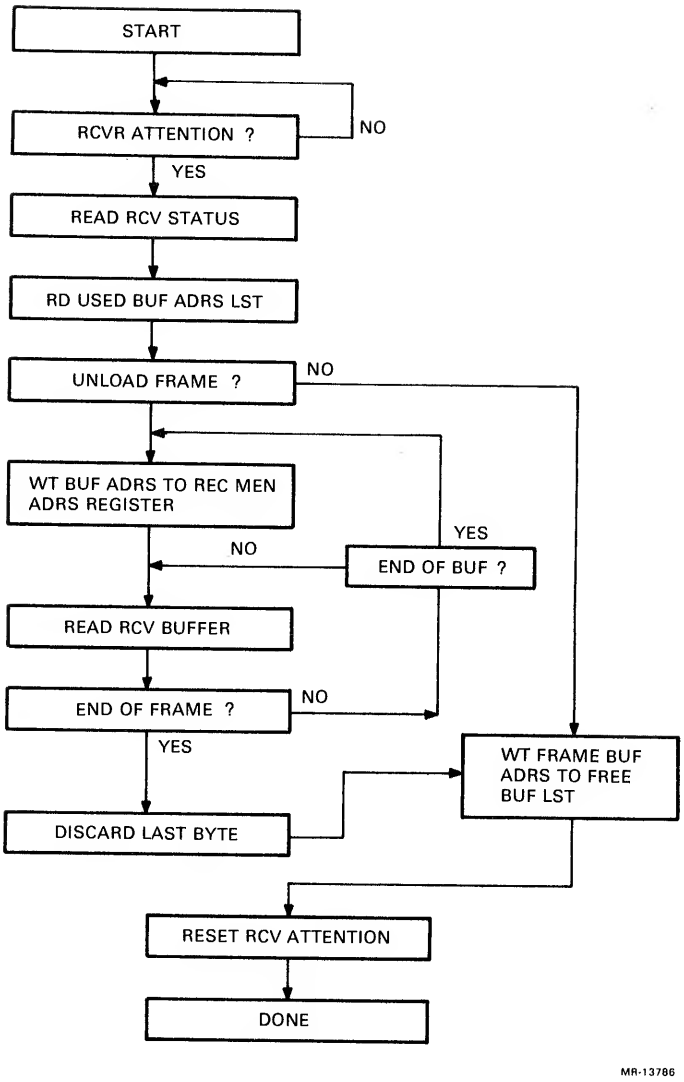


Figure 4-12 PLI Interface Receive Flow Diagram

CHAPTER 5
LOGIC DESCRIPTION

The first two sections of this chapter describe the function of the major logic elements in the three port modules and the NIA module:

1. M3001 EBus interface and port ALU module
2. M3002 port microprocessor control module
3. M3003 CBus interface and data mover module
4. L0072 Network Interconnect Adapter (NIA) module.

These major logic elements are shown on the block diagram that accompanies each module description.

Section 5.3 of this chapter describes the port functions performed by these logic elements and the microcode. It follows the organization of the port microcode, describing initialization and the idle loop, and the major functions performed out of the idle loop: interrupts, CSR pocessing, transmit packet processing, receive packet processing, queue manipulation.

5.1 EBUS INTERFACE AND PORT ALU MODULE

5.1.1 Introduction

This module provides a control and status interface (EBus interface) between the KL10 and the port microprocessor. The KL10 accesses this interface by executing DATAO, DATAI, CONO, and CONI commands. The port microprocessor accesses this interface by executing microprocessor commands. These microprocessor commands are decoded functions of microword fields MWBUSCTLFLD and MWMGCFLD. The port microprocessor monitors the status of the EBus interface by sensing condition codes (described at the end of Section 5.1.2).

The primary components of the EBus interface include

1. The control and status register (CSR). The CSR is accessed by the KL10 and the port microprocessor to pass control and status parameters from one to the other

2. A data path between the port microprocessor and the EBus. This data path includes the EMUX, MBus, EBUF, and KMUX

3. The logic required to support the EBus protocol, including the EBus interrupt sequence

4. Logic to load and start the port microcode

5. An EBus parity generator/checker

6. Logic to support various loopback and other diagnostic functions.

In addition to the EBus interface, this module contains the arithmetic and logic unit (ALU) for the port microprocessor, and a multiplexer (CNST MUX) used to enter various constants to the ALU from the control RAM (CRAM).

5.1.2 EBus Control and Status Register

The CSR is a 36-bit register used to pass control and status information between the port microprocessor and the KL10. The CSR bits are shown in Figure 5-1, and described in Table 5-1, and in the text following the table.

BIT NO.	BIT DEFINITION	RD/WR	
		KL10	PORT
00	PORT PRESENT	R	H
01	DIAG RQST CSR	R	H
02	DIAG CSR CHNG	R/H	H
03		*	*
04	RQST EXAM OR DEP	R/H	R/S
05	RQST INTERRUPT	R/H	R/S
06	CRAM PARITY ERR	R/C	H
07	MBUS ERROR	R	H
08		*	*
09		*	*
10		*	*
11	IDLE	R	R/W
12	DISABLE COMPLETE	R	R/W
13	ENABLE COMPLETE	R	R/W
14		*	*
15	PORT ID CODE 00	R	H
16	PORT ID CODE 01	R	H
17	PORT ID CODE 02	R	H

BIT NO.	BIT DEFINITION	RD/WR	
		KL10	PORT
18	CLEAR PORT	W	*
19	DIAG TEST EBUF	R/W	*
20	DIAG GEN EBUS PE	R/W	*
21	DIAG SEL LAR	R/W	*
22	DIAG SINGLE CYC	R/W	*
23	SPARE	R/W	*
24	EBUS PARITY ERR	H/R/C	R/H
25	FREE QUEUE ERR	R/C	R/S
26	DATA PATH ERR	R/C	R/S
27	CMD QUEUE AVAIL	R/S	R/C
28	RSP QUEUE AVAIL	R/C	R/S
29		*	*
30	DISABLE	R/S	R/C
31	ENABLE	R/S	R/C
32	MPROC RUN	R/W	R/H
33	PIA 00	R/W	R
34	PIA 01	R/W	R
35	PIA 02	R/W	R

* = NOT DEFINED
R = READABLE
W = WRITEABLE (SET OR CLEAR)
C = CLEARABLE ONLY
S = SETTABLE ONLY
H = HARDWARE CONTROLLED

MR-13756

Figure 5-1 CSR Bits

Table 5-1 CSR Bit Description

Bit Name Number	Function
PORT PRESENT CSR00	Indicates that the port is present. The KL10 always reads this bit as 1 if the port is present (installed and powered-up). The port always reads this bit as 0.
DIAG RQST CSR CSR01	This diagnostic bit indicates the status of CCRQSTCSR. **Set By:** The port microprocessor requesting access to the CSR (asserting MPRQSTCSR). **Cleared by:** Port microprocessor reading the CSR (asserting MPREADCSR). Port microprocessor writing the CSR (asserting MPLOADCSR). KL10 setting CLEAR (CSR18). General EBus reset.
DIAG CSR CHNG CSR02	This diagnostic bit indicates that the contents of the CSR have changed since it was last read by the port microprocessor. **Set by:** KL10 writing the CSR (executing a CONO function). Detection of an EBUS PARITY ERROR (CSR24 set). **Cleared by:** Port microprocessor reading the CSR (asserting MPREADCSR). KL10 setting CLEAR (CSR18). General EBus reset.
UNUSED CSR03	Not used by either the port microprocessor or the KL10.
RQST EXAM OR CSR04	Used by the port microprocessor to request an EBus interrupt on PI level 00 (examine or deposit function). A PI level 00 interrupt is immediately generated when this bit is set.

Table 5-1 CSR Bit Description (Cont)

Bit Name Number	Function
	Set by: Port microprocessor requesting an EBus examine or deposit interrupt on PI level 00 (asserting MPEXORDEP). **Cleared by:** Successful completion of the examine or deposit sequence. The KL10 setting CLEAR (CSR18). General EBus reset.
RQST INTERRUPT CSR05	Used by the port microprocessor to request an EBus interrupt on PI levels 01 through 07. A PI level 01 through 07 interrupt will be immediately generated when bit is set. **Set by:** Port microprocessor requesting an EBus interrupt on PI levels 01 through 07 (asserting MPRQSTINTR). CRAM PAR ERR (CSR06 set). MBUS ERR (CSR07 set). **Cleared by:** Successful completion of the interrupt sequence. KL10 setting CLEAR (CSR18). General EBus reset.
CRAM PAR ERR CSR06	Indicates that a CRAM parity error has been detected. If this bit is set the port microprocessor is immediately halted and RQST INTERRUPT (CSR05) set. A hardware nonvectored (40 + 2n) interrupt will be forced. CRAM PAR ERR may be forced in order to halt the port microprocessor at a specific location (breakpoint). The port microprocessor cannot be restarted (CSR32 set) until this bit is cleared. **Set by:** Detection of a CRAM parity error.

Table 5-1 CSR Bit Description (Cont)

Bit Name Number	Function
	Cleared by: KL10 writing a 1 in EBus PARITY ERR (CSR24). KL10 setting CLEAR (CSR18). General EBus reset.
MBUS ERR CSR07	Indicates that more than one MBus driver has been turned on at the same time. That is, more than one set of port logic is trying to drive the MBus at the same time. If this bit is set, the port microprocessor is immediately halted and RQST INTERRUPT (CSR05) set. A hardware nonvectored (40 + 2n) interrupt will be forced. The port microprocessor cannot be restarted (CSR32 set) until this bit is cleared. **Set by:** The detection of more than one MBus driver being on at the same time. **Cleared by:** The KL10 setting CLEAR (CSR18). General EBus reset.
UNUSED CSR08	Not used by either the port microprocessor or the KL10.
UNUSED CSR09	Not used by either the port microprocessor or the KL10.
UNUSED CSR10	Not used by either the port microprocessor or the KL10.
IDLE CSR11	Indicates that the port microprocessor is in the idle loop, and is not hung in some other microcode routine. Useful for debugging and troubleshooting **Set by:** Port writing a 1 in the bit.

Table 5-1 CSR Bit Description (Cont)

Bit Name Number	Function
	Cleared by: Port writing a 0 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
DISABLE COMPLETE CSR12	This microcode(software)-defined bit is used by the port to inform the KL10 operating system that the port microprocessor has placed itself in the disabled state. Set by: Port writing a 1 in the bit. Cleared by: Port writing a 0 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
ENABLE COMPLETE CSR13	This microcode(software)-defined bit is used by the port to inform the KL10 operating system that the port microprocessor has placed itself in the enabled state. Set by: Port writing a 1 in the bit. Cleared by: Port writing a 0 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
UNUSED CSR14	Not used by either the port CSR14 microprocessor or the KL10
PORT ID CODE 00 CSR15	Bit 00 of the 3-bit PORT IDENT CODE field. The KL10 always reads this bit as a 0 if the port is present (installed and powered-up). The port always reads this bit as a 0.
PORT ID CODE 01 CSR16	Bit 01 of the 3-bit PORT IDENT CODE field. The KL10 always reads this bit as a 1 if the port is present (installed and powered-up). The port always reads this bit as a 0.

Table 5-1 CSR Bit Description (Cont)

Bit Name Number	Function
PORT ID CODE 02 CSR17	Bit 02 of the 3-bit PORT IDENT CODE field. The KL10 always reads this bit as a 1 if the port is present (installed and powered-up). The port always reads this bit as a 0.
CLEAR PORT CSR18	When set by the KL10, this bit resets the port. The microprocessor is halted and all pertinent registers and control logic are placed in a reset state. **Set by:** KL10 writing a 1 in the bit. **Cleared by:** Bit clears itself after the reset function is completed.
DIAG TEST EBUF CSR19	This diagnostic bit enables the KL10 to do an EBus interface loopback function by loading and reading the EBUF. If the port is not running (CSR32 is reset) and CSR19 is set, then a KL10 DATAO loads EBus data into the EBUF, and DATAI places EBUF data on the EBus. **Set by:** KL10 writing a 1 in the bit. **Cleared by:** KL10 writing a 0 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
DIAG GEN EBUS PE CSR20	This diagnostic bit enables the KL10 to test the EBus parity checker by forcing it to decode an EBus parity error. When this bit is set, EBUS PAR ERR (CSR24) is also set on the same CONO, assuming no real EBus parity error. **Set by:** KL10 writing a 1 in the bit.

Table 5-1 CSR Bit Description (Cont)

Bit Name Number	Function
	Cleared by: KL10 writing a 0 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
DIAG SEL LAR CSR21	This diagnostic bit enables the KL10 to read the latch address register (LAR). If this bit is set, the port is not running (CSR32 reset), and the DIAG TEST EBUF (CSR19) is reset, then a KL10 DATAI causes the LAR contents to be asserted on EBus bits D01-D12. All other EBus bits are undefined. **Set by:** KL10 writing a 1 in the bit. **Cleared by:** KL10 writing a 0 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
DIAG SINGLE CYC CSR22	This diagnostic bit enables the port microprocessor to be single-cycled. If this bit is set and the KL10 sets MPROC RUN (CSR32), the port microprocessor will execute one microcycle and halt. MPROC RUN is cleared when the microprocessor halts. The current address to be executed is fetched from the RAM address register (RAR). The next address to be executed is stored in the LAR at the completion of the microcycle. The KL10 must read the address from the LAR and load it into the RAR before executing the next single cycle. **NOTE** **This bit must be reset for the KL10 to read and write the CRAM correctly.** **Set by:** KL10 writing a 1 in the bit.

Table 5-1 CSR Bit Description (Cont)

Bit Name Number	Function
	Cleared by: KL10 writing a 0 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
SPARE CSR23	Reserved for future software use. **Set by:** KL10 writing a 1 in the bit. **Cleared by:** KL10 writing a 0 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
EBUS PARITY ERR CSR24	When read by the KL10, this bit indicates that an EBus parity error has been detected. When written as a 1 by the KL10, this bit will clear itself and CRAM PARITY ERR (CSR06). **Set by:** The detection of an EBus parity error while the port is reading data from the EBus. **Cleared by:** KL10 writing a 1 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
FREE QUEUE ERR CSR25	This microcode(software)-defined bit is used by the port to inform the port driver software that there are no free queue entries available on the message free queue or the datagram free queue. The state of this bit has no hardware function. **Set by:** Port writing a 1 in the bit.

Table 5-1 CSR Bit Description (Cont)

Bit Name Number	Function
	Cleared by: KL10 writing a 1 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
DATA PATH ERR CSR26	This microcode(software)-defined bit is used by the port to inform the port driver software that it has detected an error in the DMA data path (including the mover/formatter). The state of this bit has no hardware function. **Set by:** Port writing a 1 in the bit. **Cleared by:** KL10 writing a 1 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
CMD QUEUE AVAIL CSR27	This microcode(software)-defined bit is used by the port driver software to inform the port that it has placed a command queue entry on a previously empty command queue. The state of this bit has no hardware function. **Set by:** KL10 writing a 1 in the bit. **Cleared by:** Port writing a 1 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
RESP QUEUE AVAIL CSR28	This microcode(software)-defined bit is used by the port to inform the port driver sofware that it has placed a response queue entry on the previously empty response queue. The state of this bit has no hardware function. **Set by:** Port writing a 1 in the bit.

Table 5-1 CSR Bit Description (Cont)

Bit Number	Name	Function
		Cleared by: KL10 writing a 1 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
	UNUSED CSR29	Not used by either the port microprocessor or the KL10.
	DISABLE CSR30	This microcode(software)-defined bit is used by the port driver software to tell the port to place itself in the disabled state (set CSR12). The state of this bit has no hardware function. Set by: KL10 writing a 1 in the bit. Cleared by: Port writing a 1 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
	ENABLE CSR31	This microcode(software)-defined bit is used by the port driver software to tell the port to place itself in the enabled state (set CSR13). The state of this bit has no hardware function. Set by: KL10 writing a 1 in the bit. Cleared by: Port writing a 1 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.

Table 5-1 CSR Bit Description (Cont)

Bit Name Number	Function
MPROC RUN CSR32	When set by the KL10, this bit causes the CRAM control register to reset and enables the port microprocessor clocks. The port will start cycling at the address contained in the RAR. The next and subsequent addresses will be fetched from the Am2910 sequencer (Y-outputs). **Set by:** KL10 writing a 1 in the bit. **Cleared by:** KL10 writing a 0 in the bit. KL10 setting CLEAR (CSR18). After each microword cycle if DIAG SINGLE CYC (CSR22) is set. CRAM PAR ERR (CSR06) or MBUS ERR (CSR07) setting. General EBus reset.
PIA00 CSR33	Bit 0 of the 3-bit KL10 EBus PIA field (PI 01 through 07). **Set by:** KL10 writing a 1 in the bit. **Cleared by:** KL10 writing a 0 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
PIA01 CSR34	Bit 01 of the 3-bit KL10 EBus PIA field (PI level 01 through 07). **Set by:** KL10 writing a 1 in the bit. **Cleared by:** KL10 writing a 0 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.
PIA02 CSR35	Bit 02 of the 3-bit KL10 EBus PIA field (PI level 01 through 07).

Table 5-1 CSR Bit Description (Cont)

Bit Name Number	Function
	Set by: KL10 writing a 1 in the bit. Cleared by: KL10 writing a 0 in the bit. KL10 setting CLEAR (CSR18). General EBus reset.

The KL10 accesses the CSR by executing CONO and CONI commands. The port microprocessor accesses the CSR in the sequence described in Section 5.1.3.2.

The CSR is read/write interlocked to prevent the port and the KL10 from accessing it at the same time. This interlock is condition code grant CSR (CCGRNTCSR). When the port wants to access the CSR, it executes a microprocessor request CSR (MPRQSTCSR) command. If the register is available, CCGRNTCSR is asserted by the EBus interface logic. If the CSR is not available because the KL10 is currently accessing the register with a CONI or a CONO, CCGRNTCSR will not be asserted until the CONI or CONO function is complete. The port microprocessor must wait until it senses CCGRNTCSR asserted before it attempts to access the CSR register. In the same way, if the port microprocessor is accessing the CSR when the KL10 executes a CONI or CONO, the EBus interface logic will cause the command to wait until the port access is completed. Contesting conditions between the port and the KL10 are prevented by granting access to the KL10 at CLK1 time and to the port (by asserting CCGRNTCSR) at CLK3 time.

The port microprocessor (in the MPROC RUN state, CSR32 set) accesses the CSR by executing the following sequence.

1. The port microcode first checks condition code interrupt active (CCINTRACTIVE). If it is asserted, the microcode waits until it is de-asserted before continuing.

2. To write the CSR, the port microprocessor then executes a microprocessor load EBUF (MPLOADEBUF) command to load the CSR data into the EBUF. On the same microcycle it executes an MPRQSTCSR command.

To read the CSR, the port microprocessor executes an MPRQSTCSR command.

3. The port microprocessor then checks for CCGRNTCSR.

4. When CCGRNTCSR is valid, the port microprocessor executes either a microprocessor load CSR (MPLOADCSR) or microprocessor read CSR (MPREADCSR) command.

   a. If an MPLOADCSR command is executed, the contents of the EBUF are strobed into the CSR at CLK3 time.

   b. If an MPREADCSR command is executed, the contents of the CSR are asserted on the MBus. On the same microcycle, the port microprocessor strobes the MBus data into location T0 of the Am2901 internal RAM.

5.1.3 EBus Control Logic

The EBus control logic arbitrates the EBus protocol and the port microprocessor protocol for interfacing to the EBus, and performs synchronization between the two. Figure 4-3 shows the EBus signals listed and described in Table 4-2.

5.1.3.1 Port Microprocessor Not Running -- The KL10 has full control of the port only when the port microprocessor is not running (MPROC RUN, CSR32 is reset). With the port in this state, the primary functions of the KL10 are to:

1. Load and read/verify the port microcode

2. Set up the correct initial CSR functions

3. Check for error conditions if the port is in this state because of an unexpected halt.

With the port in this state, the KL10 can also perform secondary diagnostic functions, such as write and read/verify the EBUF, generate bad parity, and single-cycle the port.

The KL10 performs these functions by executing CONOs, CONIs, DATAOs, and DATAIs. The port's EBus interface processes these functions via the normal EBus protocol. The EBus functions are described in Table 5-2.

Table 5-2 EBus Functions

Function	Description
LOAD RAR	If the KL10 executes a DATAO with bit 00 = 1, a DATOLOADRAR signal is generated. This signal causes EBus bits D01-D13 to be loaded (via the MBus) into the port RAR (located on the microprocessor control module).

Table 5-2 EBus Functions (Cont)

Function	Description
LOAD MICROWORD	If the KL10 executes a DATAO with bit 00 = 0, a DATOLOADMW signal is generated. This signal causes the 28 least significant bits of the EBus to be loaded (via the MBus) into the selected half of the port CRAM location specified by the contents of the RAR.
READ MICROWORD	If the KL10 executes a DATAI and CSR21 = 0 (not DIAG SEL LAR), a DATIREADMW signal is generated. This signal causes the contents of the selected half of the port CRAM location (specified by the contents of the RAR) to be placed on the EBus.
READ LAR	If the KL10 executes a DATAI and CSR21 = 1 (DIAG SEL LAR), a DATIREADLAR signal is generated. This signal causes the contents of the LAR to be placed on EBus D01-D12.
LOAD CSR	If the KL10 executes a CONO, a CONOLOADCSR signal is generated. This signal causes the contents of the EBus to be loaded into all the CSR bits writable by the KL10.
READ CSR	If the KL10 executes a CONI, a CONIREADCSR signal is generated. This signal causes the contents of all the CSR bits readable by the KL10 to be placed on the EBus.
LOAD EBUF	If the KL10 executes a DATAO and CSR19 = 1 (DIAG TEST EBUF), a TESTLOADEBUF signal is generated. This signal causes EBus D00-D35 to be loaded (via the MBus) into the EBUF.
READ EBUF	If the KL10 executes a DATAI and CSR19 = 1 (DIAG TEST EBUF), a TESTREADEBUF signal is generated. This signal causes the contents of the EBUF to be placed on the EBus.

5.1.3.2 Port Microprocessor Running -- When the port microprocessor is running (MPROC RUN, CSR32 set), the KL10 can access the CSR only by executing CONO or CONI commands. With the port in this state, CONO and CONI commands (LOAD CSR and READ CSR) operate as described in Section 5.1.3.1.

If the KL10 software executes DATAO or DATAI instructions while the port is running (CSR32 set), condition code CCEBUSRQST will be asserted. This is an unexpected illegal function and is ignored by

the port. Therefore, an EBus timeout occurs because the port will not return EBus transfer. A DATAO or DATAI executed by the KL10 microcode in response to a previous port examine or deposit request is not an illegal function.

5.1.3.2.1 **EBus Interrupts** -- When the port is running, the port microprocessor controls the EBus by loading an IOP function control word (IOP word is equivalent to the API function control word) into the EBUF and simultaneously generating an EBus Interrupt. Figure 5-2 shows the format of the IOP word. The types of interrupts that may be generated by the IOP word are listed by function in Table 5-3. The hardware can generate all of the interrupts, but the port microcode currently uses only 0, 4, 5, and 7.

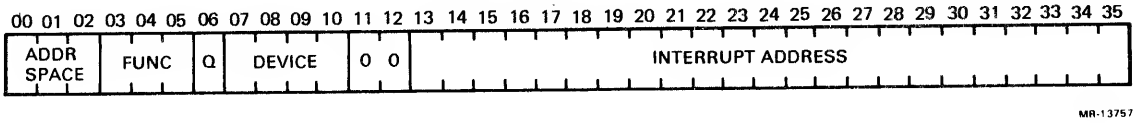


Figure 5-2 IOP Function Control Word

Table 5-3 IOP Function Control Word Bit Description

Bit	Name	Description
00-02	ADDR SPACE	The address space containing the location addressed by bits 13-35, where 0 = Executive process table (EPT) 1 = Executive virtual address space 4 = Physical memory All other codes are reserved.
03-05	FUNC	Function requested by the interrupt, where 0 = Standard (40 + 2n) interrupt (see Note) 1 = NOT USED 2 = NOT USED 3 = NOT USED 4 = DATAO (Examine) 5 = DATAI (Deposit) 6 = NOT USED 7 = DATAO (Examine and Increment). This formerly reserved function enables the CI20 to manipulate queue interlocks.

Table 5-3 IOP Function Control Word Bit Description (Cont)

Bit	Name	Description
06	Q	A qualifier interpreted according to the function code, as follows: Function Interpretation 0,7 Ignored 4,5 Q = 1, apply protection and relocation to the address specified by bits 14-35
07-10	DEVICE	Physical device number assigned by the PI system.
11-12	00	
12-35	INTERRUPT ADDRESS	The address where interrupt handling begins.

NOTE

(40 + 2n) means EPT location (40 + 2n), where n is the PI level number. For example, level 3 interrupts would reference EPT location 46. The IOP word for Function 0 interrupts is all zeros.

If the EBus interrupt is an examine or deposit request (function 4, 5, or 7), the port microprocessor requests the interrupt on PI level 00 by executing the microprocessor examine or deposit command (MPEXORDEP). PI level 00 examine/deposit interrupt requests are always processed by the KL10 (as highest priority), because PI level 00 interrupts cannot be selectively enabled or disabled as can interrupts on PI levels 01 through 07. Therefore, the KL10 will process CI20 examine and deposit requests regardless of the enable or disable conditions of the KL10 PI system.

If the EBus interrupt is a function 0, the port microprocessor requests the interrupt on PI level 01 through 07 (as assigned in CSR33-35), by executing the microprocessor request interrupt (MPRQSTINTR) command.

The interrupt sequence is given in the following 10 steps:

1. The port microcode first checks condition code interrupt active (CCINTRACTIVE). If it is asserted, the microcode must wait until the condition code is de-asserted before continuing.

2. The port microcode then builds an IOP function control word and loads it into the EBUF with a load EBUF (MPLOADEBUF) command. With the same microword, it executes a request examine or deposit (MPEXORDEP) command or a request interrupt (MPRQSTINTR) command.

   The basic IOP words (less the interrupt address) for MPEXORDEP, functions 4, 5, and 7 are predefined in local storage RAM.

   An MPRQSTINTR, function 0 is a standard (40 + 2n) interrupt, and the IOP word is all 0s.

3. MPEXORDEP or MPRQSTINTR

   The MPEXORDEP command causes RQST EXAM OR DEP (CSR04) to be set. This in turn causes the port EBus interface to assert EBus PI request line PI00.

   The MPRQSTINTR command causes RQST INTERRUPT (CSR05) to be set. This in turn causes the port EBus interface to assert the EBus PI request line (PI01-PI07) specified by the port PI level (PIA00-PIA02, CSR33-35).

4. When the KL10 EBOX recognizes the PI request it responds by asserting the following:

   The port channel number on EBus CS04-CS06
   PI SERVED (100) on EBus F00-F02
   EBus DEMAND, after delay determined by the KL10.

5. The port EBus interface responds by asserting the EBus data line corresponding to the port physical device number. (EBus D07 for RH20 position 7, the CI20 position, or EBus D05 for RH20 position 5, the NI port position.)

6. After a KL10-determined delay, the KL10 EBOX reads the EBus data lines and negates EBus DEMAND.

7. Next, the KL10 EBOX asserts:

   The port channel number on EBus CS04-CS06
   The port physical device number on EBus CS00-CS03
   PI ADR IN (101) on EBus F00-F02
   EBus DEMAND, after a KL10-determined delay

8. The port EBus interface responds by asserting:

   EBus ACKN (acknowledge).
   The IOP function control word (previously loaded into EBUF by the port microprocessor) on EBus D00-D35.
   EBus XFER (transfer), after a port-determined delay.

9. When the KL10 EBOX detects EBus XFER, it strobes the data from the EBus data lines and negates EBus DEMAND.

10. The trailing edge of EBus DEMAND causes the port EBus interface to negate EBus ACKN, EBus XFER, and the EBus data lines.

5.1.3.2.2 Examine/Deposit Request Response -- The KL10 microcode decodes the IOP function control word and executes the appropriate function. If the IOP word specifies a port examine or deposit request, then the first EBus cycle following the IOP word read will be a DATAO or DATAI to the port. The sequence is outlined in the following five steps:

1. The KL10 EBOX asserts:

   A device code of zero (0000000) on EBus CS00-CS06
   DATAO or DATAI (010 or 011) on the EBus F00-F02
   Data on the EBus data lines, if a DATAO function
   DEMAND, after a KL10-determined delay.

2. The port EBus interface responds by asserting EBus ACKN. It also flags the port microprocessor by asserting condition code EBus request (CCEBUSRQST).

   The device code returned by the KL10 on EBus CS00-CS06 is zero, not the port device code, and is ignored by the port. The port does not examine the device code because it expects that this EBus cycle is in response to its examine or deposit request. Therefore, as soon as the port senses EBus DEMAND, it does the following:

   Upon detecting CCEBUSRQST the port microcode executes either a microprocessor load EBus (MPLOADEBUS) command in response to a DATAI, or a microprocessor read EBus (MPREADEBUS) command in response to a DATAO.

   If the port microcode executes MPLOADEBUS, it must have previously executed MPLOADEBUF to load the EBUF with valid data for transfer to the KL10.

   If the port microcode executes MPREADEBUS, the EBus data is placed on the MBus. On the same microcycle, the port microcode strobes the data from the MBus into one of its internal storage media.

3. After a port-determined delay, the port EBus interface asserts EBus XFER.

4. When the KL10 EBOX detects EBus XFER it negates EBus DEMAND. If the function was a deposit (DATAI), it strobes the data from the EBus data lines. If the function was an examine (DATAO), it de-asserts the data from the EBus data lines.

5. The trailing edge of EBus DEMAND causes the port EBus interface to negate EBus ACKN, EBus XFER, CCEBUSRQST, and the EBus data lines (if a DATAI function).

For examine/deposit functions, the port microcode responds promptly to CCEBUSRQST, in order to prevent EBus timeouts. The port microprocessor does not attempt to execute any additional EBus transfers until it detects the negation of CCEBUSRQST.

5.1.4 Microprocessor to EBus Register

The microprocessor to EBus register (EBUF) is a 36-bit register normally used by the port microprocessor to pass data from the MBus (internal tristate microprocessor bus) to either the EBus or the CSR. The port microprocessor usually loads data into the EBUF from the MBus. On the next microcycle, the data is strobed from the EBUF to either the CSR or the EBus. The major functions of the EBUF are as follows:

1. To load the CSR. Data is first strobed into the EBUF, then to the CSR.

2. To transmit an IOP word over the EBus for PI level 00 examine/deposit requests.

3. To transmit an IOP word over the EBus for PI levels 01-07 vectored and nonvectored (40 + 2n) interrupt requests.

   Once the IOP word for a PI level 01-07 interrupt request has been loaded into the EBUF, the port microcode monitors condition code interrupt active (CCINTRACTIVE). The port microprocessor does not load other data into the EBUF until CCINTRACTIVE is de-asserted.

Data currently on the MBus is loaded into the EBUF at CLK3 time with a microprocessor load EBUF (MPLOADEBUF) command.

A diagnostic loopback path, controlled by DIAG TEST EBUF (CSR19), enables the KL10 to load and read the EBUF. If CSR19 is set and the port microprocessor is not running (MPROC RUN, CSR32 reset):

1. A DATAO executed by the KL10 causes EBus data to be loaded (via the MBus) into the EBUF.

2. A DATAI executed by the KL10 causes the data in the EBUF to be asserted on the EBus.

5.1.5 EBus to Microprocessor Multiplexer (EMUX)

The EMUX is a two-input by 36-bit multiplexer that passes data to the MBus from either the EBus or the CSR. The EMUX is enabled by the port microprocessor or KL10 DATAO functions. Figure 5-3 is a simplified diagram of EMUX control and data flow.

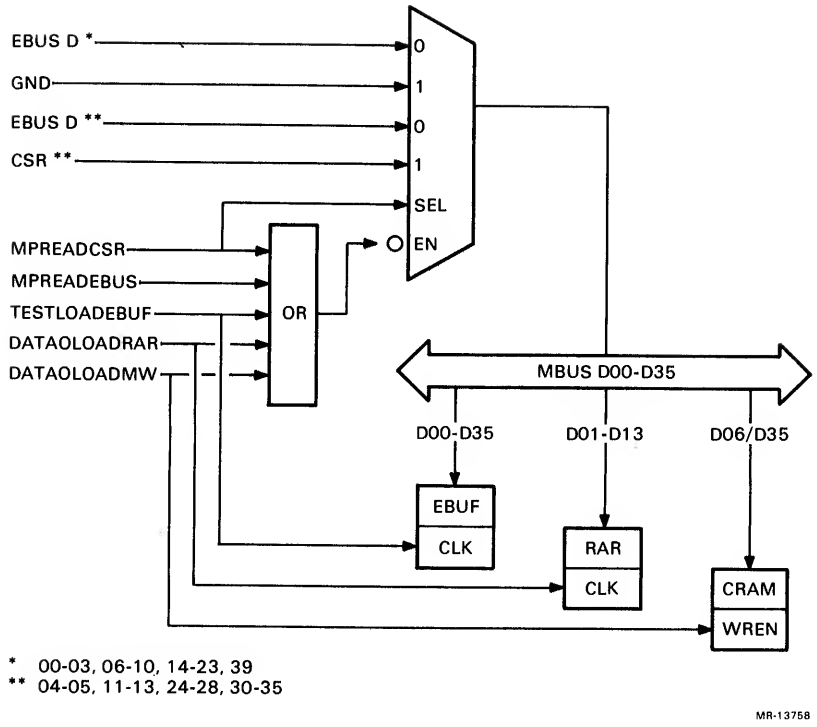


Figure 5-3 EBus to Microprocessor Multiplexer

When the port is running (MPROC RUN, CSR32 set), the EMUX is enabled with either a microprocessor read EBus (MPREADEBUS) command or a microprocessor read CSR (MPREADCSR) command. The MPREADCSR command also selects the CSR input to the EMUX. The port microprocessor can then strobe the data from the MBus into location T0 of the Am2901 internal RAM.

When the port is not running (MPROC RUN, CSR32 reset) the KL10 can enable the EMUX by executing DATAO commands to write the following:

1. The CRAM (DATAOLOADMW)
2. The RAR (DATAOLOADRAR)
3. The E buffer via the diagnostic loopback path (TESTLOADEBUF).

NOTE
CSR input to the EMUX is not selectable with KL10 DATAO commands.

5.1.6 Microprocessor to EBus Multiplexer (KMUX)

The KMUX is a two-input by 36-bit multiplexer that passes data to the EBus from either the E buffer or the CSR. The KMUX is enabled by either the port microprocessor or KL10 DATAI or CONI commands. Figure 5-4 is a simplified diagram of KMUX control and data flow.

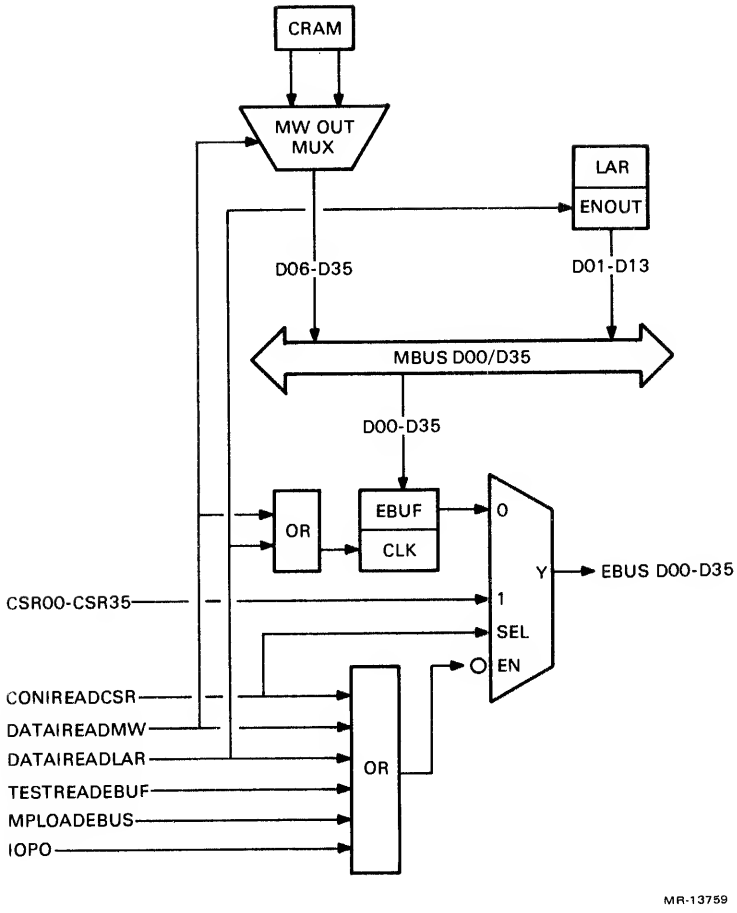


Figure 5-4 Microprocessor to EBus Multiplexer

When the port is running (MPROC RUN, CSR32 set) the KL10 can enable the KMUX to read the following:

The CSR, by executing CONI commands (CONIREADCSR)

The IOP word, by asserting PI ADR IN (101) on EBus F00-F02 during an interrupt sequence.

The port microprocessor can enable the KMUX to pass MBus data to the EBus by executing a microprocessor load EBus (MPLOADEBUS) command.

When the port is not running (MPROC RUN, CSR32 reset) the KL10 can enable the KMUX, by executing DATAI or CONI commands, to read the following:

CRAM (DATIREADMW)
LAR (DATIREADLAR)
EBUF (TESTREADEBUF)
CSR (CONIREADCSR).

5.1.7 EBus Parity Generator

The EBus parity generator generates odd parity on every 36-bit data word that the port passes to the EBus. Because the KL10 architecture prohibits parity checking on an IOP word, the signals EBUS PARITY and EBUS PARITY ACTIVE are inhibited when an IOP word is transmitted on the EBus.

5.1.8 EBus Parity Checker

The EBus parity checker normally checks for odd parity on every 36-bit data word that the port reads from the EBus. If parity is incorrect, the EBUS PARITY ERROR bit (CSR24) is set.

However, if DIAG GEN EBUS PE (CSR20) is set, the EBus parity checker checks for even parity. Therefore (assuming normal odd parity is correct), the same CONO that writes CSR20 will also cause CSR24 to be set.

5.1.9 EBus Transceivers

The EBus transceivers are the same type 8838 open collector transceivers used by other devices that interface to the EBus.

5.1.10 Arithmetic Logic Unit

The ALU is logically part of the port microprocessor and is located on the EBus interface/port ALU module. The ALU consists of

1. Nine type Am2901 four-bit bipolar microprocessor slices
2. Four type Am2902 high-speed look-ahead carry generators
3. Five type 74LS157 multiplexers used to input constants to the ALU

The nine Am2901s and four Am2902s are connnected in a parallel configuration, forming a 36-bit ALU with high-speed carry look-ahead. Figure 5-5 is a simplified diagram of the nine Am2901s.

The outpput of the ALU is connected directly to the MBus (see Figure 5-5). The data input is from the constant multiplexer (described below) and the MBus or both.

CPUCLOCK (CLK4 gated by MPROC RUN, CSR 32) is the clock input to the ALU.

For shift operations, zeros are always shifted into either the MSB or the LSB, depending on the direction of the shift.

The port microprocessor controls the ALU by executing the commands described in Table 5-4.

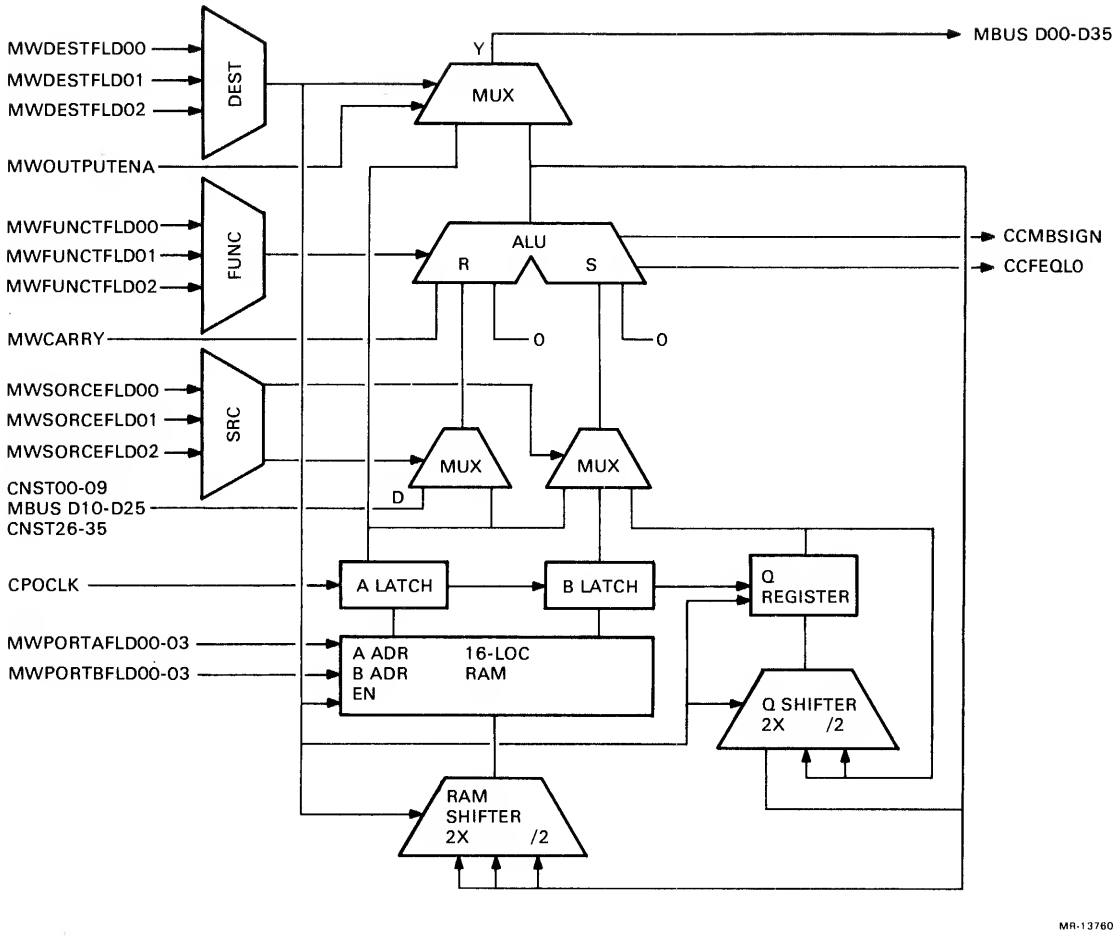


Figure 5-5 AM2901 ALU Block Diagram (Simplified)

Table 5-4 ALU Control Commands

Command	Description
MWSORCEFLD 00-02	The microword ALU source field, bits 24-26, selects the source of the R and S inputs to the ALU, as follows:

00-02	R	S
0	A	Q
1	A	B
2	Z	Q
3	Z	B
4	Z	A
5	D	A
6	D	Q
7	D	Z

Table 5-4 ALU Control Commands (Cont)

Command	Description
	Where: A = The contents of the RAM location addressed by MWPORTAFLD 00-03 B = The contents of the RAM location addressed by MWPORTBFLD 00-03 D = The data on CNST 00-09, MBus 10-25, and CONST 26-35 Q = The contents of the Q register Z = Zero
MWFUNCTFLD 00-02	The microword ALU function field, bits 27-29, controls the functions that the ALU performs on the R and S inputs, as follows: 00-02 Function 0 R plus S 1 S minus R 2 R minus S 3 R or S 4 R and S 5 -R and S 6 R xor S 7 R xnor S
MWDESTFLD	The microword ALU destination field, bits 30-32, determines if the Am2901 output (Y) will be from the ALU or the RAM location addressed by MWPORTAFLD 0-3. It also determines the input to the Q register and to the RAM location addressed by MWPORTBFLD 0-3. This field also controls the RAM shifter and Q shifter, causing the RAM and Q register inputs to multiplied or divided by 2 (shifted left or right). Thus, by controlling the Am2901 internal data paths, this field determines the destinations of the Am2901 internal data. The effect of this field is as follows: Y Q RAM B 00-02 Gets Gets Gets 0 ALU ALU HOLD 1 ALU HOLD HOLD 2 RAM A HOLD ALU

Table 5-4 ALU Control Commands (Cont)

Command	Description			
	3	ALU	HOLD	ALU]
	4	ALU	Q/2	ALU/2
	5	ALU	HOLD	ALU/2
	6	ALU	QX2	ALUX2
	7	ALU	HOLD	ALUX2
MWPORTAFLD 00-03	The microword port A select field, bits 35-38, addresses one of 16 RAM locations that will be read through the A latch. This port is read-only.			
MWPORTBFLD 00-03	The microword Port B select field, bits 39-42, addresses one of 16 RAM locations that will be either read through the B latch or written.			
MWCARRY	The microword 2901 carry in field, bit 51, is the carry into the least significant bit of the ALU. When MWCARRY is 0, zero is carried into the ALU LSB. When MWCARRY is 1, one is carried into the ALU LSB.			
MWOUTPUTENA	The microword output enable to 2901, bit 13, enables the output of the ALU.			

The port microprocessor monitors the ALU status by sensing the following condition codes:

CCFEQL0 Condition Code F = 0 -- indicates that the result of the last ALU operation produced all zeros

CCMBSIGN Condition Code MBus Sign -- indicates that the last ALU operation set the sign bit (ALU MSB = MBus bit 00)

5.1.11 Constant Multiplexer

The constant multiplexer (CNST MUX) is a two-input by 20-bit multiplexer that provides the data (D) inputs to the 10 LSBs and 10 MSBs of the Am2901s (see Figure 5-6). It passes either MBus D00-D09 or MWMGCFLD 00-09 (microword magic number, bits 14-23) to the MSBs, and either MBus D26-D35 or MWMGCFLD 00-09 to the LSBs. This allows the microprocessor to load a constant number value into the 10 most significant and 10 least significant bits of the ALU. MBus D10-D25 are always loaded into the corresponding "D" inputs.

MWMGCFLD 00-09 are selected as input to CNST MUX 00-09 and 26-35 when the port microprocessor executes a microword with the microword skip/condition (MWSKIPFLD), bits 43-47, set to 24 (octal). This SKIP/COND field value causes the signal select constant field (SELCNSTFLD) to be asserted.

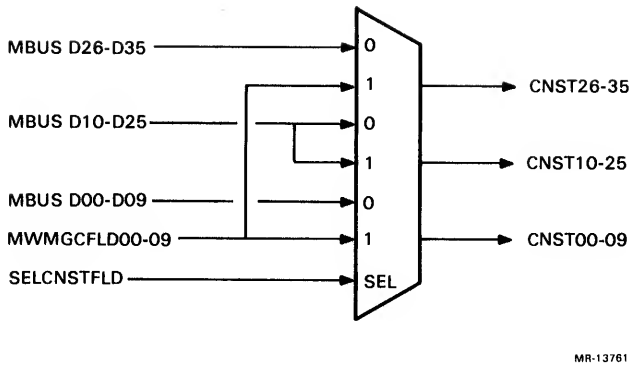


Figure 5-6 Constant Multiplexer (Simplified)

5.2 CBUS/DATA MOVER (CMVR) INTERFACE MODULE

The CBus/data mover (CMVR) interface module consists primarily of the following control logic and data paths.

1. The CMVR control logic decodes and executes the commands specified by the microprocessor controller microword. The port microprocessor accesses the CMVR module by executing microprocessor commands. These commands are decoded functions of the microword bus control (MWBUSCTLFLD) field, bits 48-50 and the MWMGCFLD field of the CRAM control word. The port microprocessor monitors the CMVR control logic status by sensing condition codes.

2. A data path between the KL10 CBus and the packet buffer port link interface (PLI), including:

   A data mover and formatter (MVR/FMTR) between the CBus and the PLI. The mover and formatter maps PLI 8-bit bytes into KL10 36-bit words and KL10 36-bit words into PLI 8-bit bytes.

   CBus input and output buffers, CBus parity generators and checkers, and CBus control logic.

   PLI input and output buffers, PLI parity generators and checkers, and PLI control logic.

3. A data path between the CMVR module and the port microprocessor. This data path enables the microprocessor to:

   Load or read mover and formatter
   Load or read the packet buffers via the PLI.

4. A parity predictor for checking parity through the mover and formatter.

5.2.1 CMVR Control Logic

The CMVR control logic consists primarily of a decoder and a series of two-input NAND gates. The decoder input is microword bus control (MWBUSCTLFLD 00-02), bits 48-50. Table 5-5 lists the decoder output signals as follows:

Table 5-5 Decoder Output Signals

MWBUSCTLFLD 0-2	Decoder Output
000	nc (not connected)
001	SELPLIFLD (select PLI field)
010	SELMBUSFLD (select MBus field)
011	SELFMTRFLD (select mover and formatter field)
100	SELSCUSFLD (select CBus field)
101-111	nc (not connected)

These decoder output signals are then ANDed with various combinations of microword magic number (MWMGCFLD 02-09), bits 16-23 to control the PLI, CBus, and MBus interfaces, and the mover and formatter.

The CMVR control logic also generates and distributes the port clocks (CLK1, CLK2, CLK3 and CLK4) to all three port modules. All four clocks make up a single microcycle. The port clocks are derived from the KL10 EBus clock. Figure 5-7 shows the relationship of the clocks when the KL10 EBus clock is running at the normal 160 ns cycle time.

CLK1 normally strobes the next microword into the port CRAM control register, and has several other timing functions, depending on the specific operation. CLK2, CLK3, and CLK4 are generally used throughout the port control logic to execute microword-specified functions. CLK4 generates CPUCLOCK, which is the clock input to the Am2901 ALU.

All four clocks are gated by control logic on the port microprocessor control module. The gated clocks are named RUNCLK1, RUNCLK2, RUNCLK3, and RUNCLK4, respectively. Gating these clocks provides orderly control of port starting, stopping, and single-cycle operation.

When the microword time field (MWTIMEFLD), bit 56, is set in a microword, CLK3 and CLK4 will occur 160 ns later than normal for that microcycle, and the microcycle time will increase from 320 ns to 480 ns. Increasing the execution time of the microcycle is a simple way to overcome specific microinstruction timing problems.

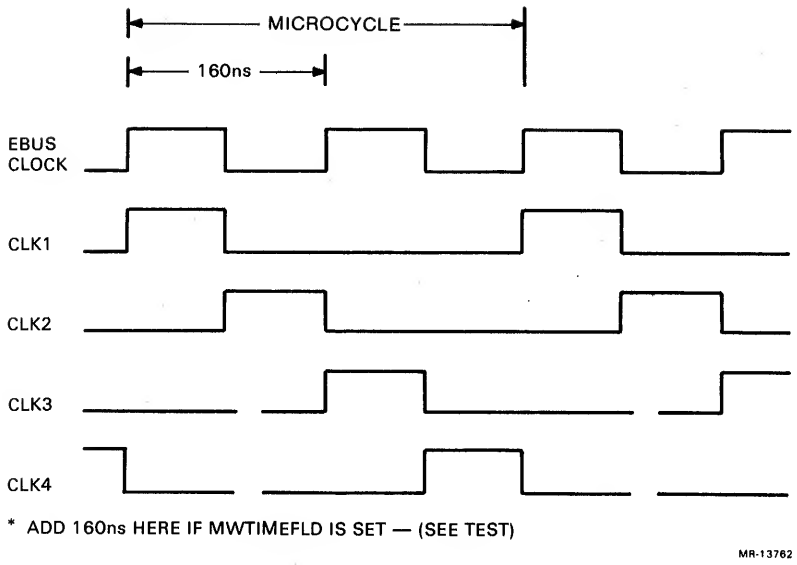


Figure 5-7 Port Clock Timing

5.2.2 Data Mover and Formatter (MVR/FMTR)

The mover/formatter consists of four parallel/serial shift registers and associated control logic. It is used as follows:

1. Parallel load and read by the port microprocessor as a 36-bit register.

2. Parallel load and read by the CBus interface as a 36-bit register.

3. Parallel read by the PLI as an 8-bit register.

4. Serial load and left shift (shift up, from LSB to MSB) by the PLI, four or eight bits at a time.

5. Serial load and right shift (shift down, from MSB to LSB) by the PLI, four or eight bits at a time.

Four-bit nibbles from the PLI are loaded into the mover/formatter and shifted either up (LSB to MSB) or down (MSB to LSB), to form 36-bit KL10 words for parallel transfer to the CBus. Two shifts load one 8-bit byte. One or two shifts can be executed in a single port microprocessor microcycle.

The 36-bit KL10 words are loaded from the CBus into the mover/formatter. After a word is loaded, the port microprocessor can shift the word, up or down, into the PLI output register for transfer to the PLI.

If the PLI output register is loaded from the bottom of the mover/formatter and shifted up (LSB to MSB) the data is not wrapped around, but is shifted out of the four MSBs of the mover/formatter and lost. If the PLI output register is loaded from the top and shifted down (MSB to LSB) the data can be wrapped around. That is, the four LSBs of the mover/formatter are input to the four MSBs of the mover/formatter. Therefore, data can be right-shifted around the mover/formatter as necessary, to align the data into the proper format. The port microprocessor can select either the four MSBs or the four LSBs of the PLI byte as the first input to the mover/formatter.

The commands to control the mover/formatter are described in Table 5-6.

Table 5-6 Mover/Formatter Control Commands

Command	Description
MPCBUFTOFMTR	C bufffer to formatter -- causes the C buffer contents to be loaded into the mover/formatter registers.
MPFMTRTOPLOUT	Formatter to PLI output buffer -- causes the 8-bit data byte in the mover/formatter PLI output register to be loaded into the PLI output buffer.
MPZEROLFTNIB	Zero left nibble -- when asserted, causes the four MSBs of the mover/formatter PLI output register to be forced to zeros before they are loaded into the PLI output buffer.
MPRHTNIBFIRST	Right nibble first -- when asserted, the first bits shifted into the mover/formatter are the four PLI input buffer LSBs. When not asserted, the first bits shifted into the mover/formatter are the four PLI input buffer MSBs.
MPSHFTFMTR8	Shift formatter by 8 bits -- causes the mover/formatter contents to be shifted eight bits to the left or right, depending on the state of MPSHIFTRIGHT.
MPSHFTFMTR4A	Shift Formatter by 4 Bits -- causes the mover/formatter contents to be shifted four bits to the left or right, depending on the state of MPSHIFTRIGHT.

Table 5-6 Mover/Formatter Control Commands (Cont)

Command	Description
MPSHFTFMTR4B	Shift Formatter by 4 Bits -- causes the mover/formatter contents to be shifted four bits to the left or right, depending on the state of MPSHIFTRIGHT.
MPSHIFTRIGHT	Shift right -- when asserted, either the currently selected PLI input buffer nibble or the four mover/formatter LSBs (MVROUT 36-39) are shifted into the four mover/formatter MSBs and shifted right. The data shifted into the mover/formatter is selected by MPPLINTOFMTR. If both MPSHIFTRIGHT and MPPLINTOFMTR are asserted, the currently selected PLI input buffer nibble is shifted into the four mover/formatter MSBs and shifted right. If MPSHIFTRIGHT is asserted and MPPLINTOFMTR is not asserted, then MVROUT 36-39 is shifted into the four mover/formatter MSBs and shifted right. If MPSHIFTRIGHT is not asserted and if MPPLINTOFMTR is asserted, the PLI input buffer nibbles are shifted into the four mover/formatter LSBs and shifted left.
MPPLINTOFMTR	PLI input buffer to formatter -- causes the 8-bit data byte currently in the PLI input buffer to be shifted (four bits at a time) into the mover/formatter. This command is executed in conjunction with MPSHFTFMTR4A or MPSHFTFMTR8. If MPSHFTFMTR4A is executed, then only four PLI input buffer bits are shifted into the mover/formatter. If MPSHFTFMTR8 is executed, then all eight PLI input buffer bits will be shifted into the mover/formatter. Two 4-bit shifts can be executed during one microcycle, right shifting an entire 8-bit byte in a single microcycle.

Figure 5-8 shows the flow of data through the mover/formatter, which is essentially four 12-bit shift registers. The data in and out is from or to the PLI, CBus, or MBus.

For CBus and MBus data, the mover/formatter acts as a single 36-bit register, inputting and outputting CBus and MBus data in 36-bit parallel transfers.

PLI data is input to the mover/formatter, four bits at a time. It is either loaded into the LSBs and shifted up, or into the MSBs and shifted down. Data is output to the PLI, eight bits at a time, from the two MSBs in each of the four registers.

The LSB of each of the four registers (36-39) can be wrapped around and shifted into the MSBs and down, in order to align the data.

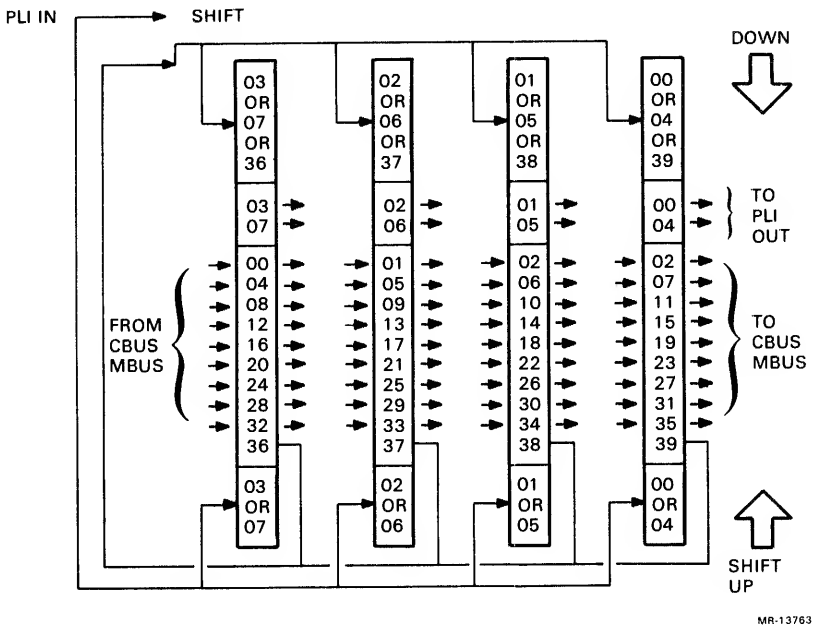


Figure 5-8 Mover/Formatter Data Flow

The mover/formatter supports three different data formats (byte packing modes):

1. Industry Compatible: Four 8-bit bytes per 36-bit word. Bits 32-35 of the word are forced to zero (Figure 2-39).

2. Core Dump: Five 8-bit bytes per 36-bit word. Four bits of every fifth byte are discarded.

3. High Density: Four and one-half 8-bit bytes per 36-bit word (Figure 2-32).

These data formats are implemented by different microcode subroutines under control of the port microprocessor.

5.2.3 Data Input Multiplexer

The DMUX is a two-input by 36-bit multiplexer that passes a 36-bit data word to the mover/formatter from either the CBus input buffer or the CBUF. When MPCBUFTOFMTR is asserted, it selects the CBUF input to DMUX; otherwise the CBus input buffer input is selected. Figure 5-9 shows data flow through the DMUX.

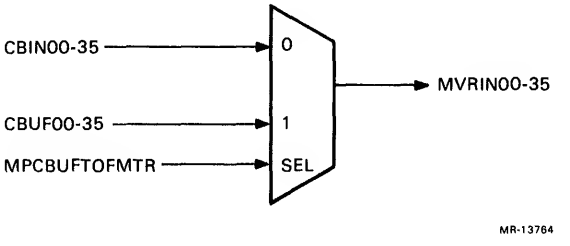


Figure 5-9 DMUX (Simplified)

5.2.4 PLI Serial Up Multiplexer (SUMUX)

The SUMUX is a two-input by 4-bit multiplexer that passes either the four LSBs or the four MSBs of the PLI input buffer to the four mover/formatter LSBs (see Figure 5-10). The 4-bit nibbles can then be left shifted to form a 36-bit word. Each mover/formatter left shift discards the four MSBs.

Each microcycle can process two 4-bit nibbles, inputting an entire 8-bit PLI input buffer byte to the mover/formatter in one microcycle.

MPPLINTOFMTR enables the SUMUX. MPRHTNIBFIRST selects the four PLI input buffer LSBs as inputs.

The nibbles from the SUMUX are also an input to the SDMUX (see Section 5.2.5).

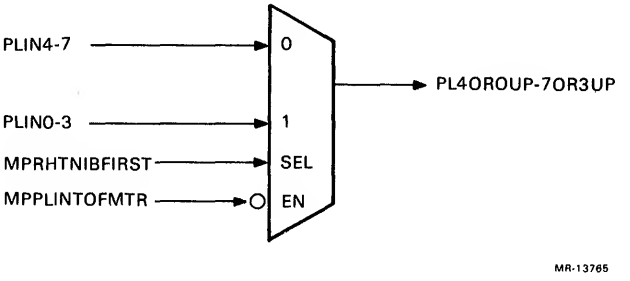


Figure 5-10 SUMUX (Simplified)

5.2.5 PLI Serial Down Multiplexer (SDMUX)

The SDMUX is a two-input by 4-bit multiplexer that passes either 4-bit SUMUX output nibbles or the four mover/formatter LSBs (MVROUT 36-39) to the four mover/formatter MSBs (see Figure 5-11). The nibbles can then be right shifted to form a 36-bit word. Each microcycle can process two 4-bit nibbles, inputting an entire 8-bit PLI input buffer byte to mover/formatter in one microcycle.

The capability to shift the four mover/formatter LSBs (MVROUT 36-39) back into back into the four MSBs (PL4OR0DN-7OR3DN) permits data to be wrapped around and shifted indefinitely with right shift commands. Therefore, data can be retained and shifted (in 4-bit nibbles) to any position in the mover/formatter registers.

MPSHIFTRIGHT enables the SDMUX, and MPPLINTOFMTR selects the output of SUMUX as input.

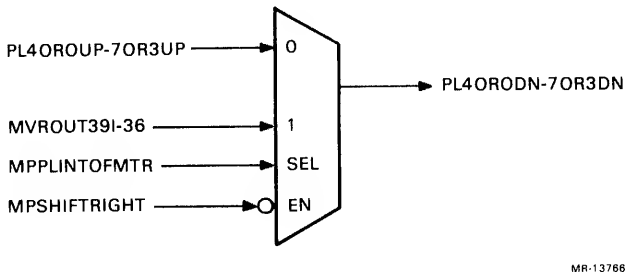


Figure 5-11 SDMUX (Simplified)

5.2.6 PLI Output Multiplexer (PMUX)

The PMUX is a two-input by 8-bit multiplexer that passes either the two MSBs from each of the four mover/formatter registers (MVRPLOUT0-7) or the eight LSBs of the CBUF to the 8-bit PLI output buffer (see Figure 5-12).

The port microprocessor command, MPCBUFTOPLOUT (CBUF to PLI output buffer) selects CBUF28-35 as the input to the PMUX, passing MBus D28-D35 (via CBUF) to the PLI output buffer. Otherwise, MVRPLOUT0-7 are passed to the PLI output buffer.

If MPZEROLFTNIB is asserted, the four MSBs of the PMUX output (PLOUT4-7) are forced to zeros, by disabling half of the PMUX. This capability is needed for core dump byte-packing mode (see Figure 5-11), where the four MSBs of every fifth byte (BYTE 5n+5) must be zero.

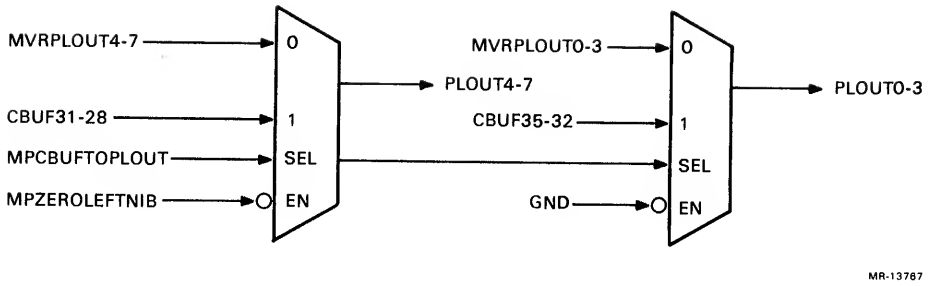


Figure 5-12 PMUX (Simplified)

5.2.7 CMVR to Microprocessor Multiplexer (CMUX)

The CMUX is a two-input by 36-bit multiplexer that interfaces to the port's internal MBus (see Figure 5-13). It enables the port microprocessor to read

The mover/formatter as a 36-bit register, with an enable CMUX (MPENACMUX) command.

The packet buffers (via the PLI input buffer) as 8-bit bytes. A PLI input buffer to CMUX (MPPLINTOCMUX) command selects the PLI input buffer (PLIN0-7) as input to the eight CMUX LSBs. The 28 CMUX MSBs are forced to 0, by tying their inputs to ground. MPENACMUX must also be asserted, to enable the multiplexer.

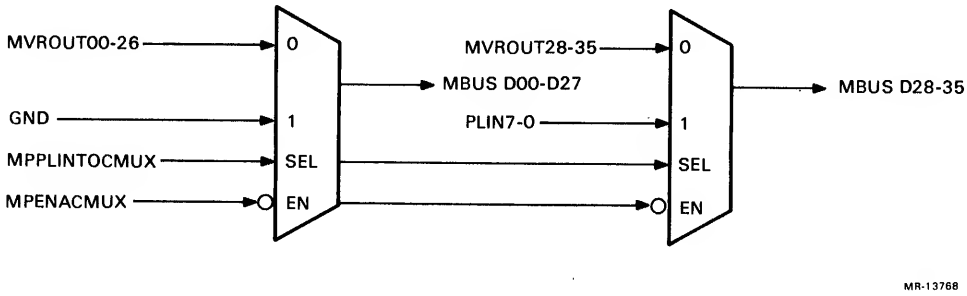


Figure 5-13 CMUX (Simplified)

5.2.8 Microprocessor to CMVR Register (CBUF)

The CBUF is a 36-bit buffer/driver that passes data from the microprocessor internal MBus to the CBus/data mover interface module (CMVR). The CBUF acts only as an isolation buffer to the tristate MBus and is logically transparent to the port microprocessor.

5.2.9 CBus Input Buffer

The CBus input buffer is a latched 38-bit (36 data bits + 2 parity bits) register that passes data from the CBus to the mover/formatter. The register can be loaded from the CBus whenever it is not being read by the mover/formatter. The reverse is also true. That is, the register can be read by the mover/formatter if it is not being loaded from the CBus.

The contents of the CBus input buffer are normally clocked into the mover/formatter by a CBus input buffer to formatter (MPCBINTOFMTR) command at CLK2 time, if condition code CBus available (CCCBUSAVAIL) was asserted on the previous microcycle.

5.2.10 CBus In Parity Checker

The CBus In parity checker checks for odd parity on each 18-bit half of the 36-bit word that the mover/formatter reads from the CBus input buffer. If parity is not correct, condition code CBus parity error (CCCBUSPARRERR) is generated and passed to the microsequencer (Am2910) condition code multiplexer (CCMUX).

The condition code remains latched until cleared by the microsequencer. When the port microprocessor senses CCCBUSPARERR is set, it sets DATA PATH ERR (CSR26) in the control and status register.

5.2.11 CBus Output Buffer

The CBus output buffer is a latched 38-bit (36 data bits + 2 parity bits) register that passes data from the mover/formatter to the CBus. The register can be loaded from the mover/formatter whenever it is not being read by the CBus. The reverse is also true. That is, the register can be read by the CBus if it is not being loaded from the mover/formatter.

The CBus output buffer is normally loaded from the mover/formatter by a formatter to CBus output buffer (MPFMTRTOCBOUT) command at CLK2 time, if CCCBUSAVAIL was asserted on the previous microcycle.

5.2.12 CBus Out Parity Generator

The CBus Out parity generator generates odd parity for each 18-bit half-word passed from the mover/formatter to the CBus output buffer. The two parity bits for a complete 36-bit word are latched into the buffer for output to the CBus.

5.2.13 CBus Control Logic

The CBus control logic arbitrates the CBus protocol and the port protocol for starting and stopping the CBus, and performs synchronization between the CBus and the mover/formatter. The CBus control logic also generates the clock timing for the port.

The CBus is a synchronous, high-speed, time-division multiplexed, tristate data bus. It runs between the KL10 MBOX and the channel devices (see Figure 4-5 and Table 4-5). Each device on the CBus has a unique time slot. A CBus data transfer has four cycles: select, request, wait, and data (see Figure 4-6 and Table 4-6).

The port microcode prepares to start the CBus data channel by executing a start CBus (MPSTARTCBUS) command. For an input data transfer to the KL10 memory, the port microcode also executes a write to KL10 memory (MPWRITEMEM) command. The CBus control logic latches these commands, until they can be executed when the port's CBus time slot becomes available. The CBus control logic detects the port's time slot by sensing its CBus SEL line.

When the CBus control logic detects the port's CBus SEL line asserted and the CBus READY line negated, it starts the channel by using the latched MPSTARTCBUS command to assert CBus START and CBus RESET during the subsequent data cycle. The CBus control logic then clears the appropriate latches set by previous port microcode commands (such as MPSTARTCBUS). If the transfer is to KL10 memory, the CBus control logic also uses the MPWRITEMEM command to assert CBus CTOM at this time. However, it does not clear the latch set by the MPWRITEMEM command until the data transfer is complete.

When the KL10 channel is ready to transfer data over the CBus, it asserts CBus READY during the port data cycle.

After receiving CBus READY, the port CBus Control asserts CBus REQUEST during its request cycle whenever it requires a data word from the channel (device write), or whenever it requires that the channel accept a data word (device read). The words are asserted on the CBus DATA lines during the port data cycle following its corresponding request cycle.

The port is ready to transfer data across the CBus whenever its CBus input buffer is empty, or whenever its CBus output buffer is full.

The CBus input buffer is emptied (transferred to the mover/formatter) with the CBus input buffer to formatter (MPCBINTOFMTR) command when the port microprocessor senses the condition code CBus available (CCCBUSAVAIL) and is prepared to accept data from the CBus.

The CBus output buffer is loaded (the contents of the mover/formatter are transferred to it) with a formatter to CBus output buffer (MPFMTRTOCBOUT) command when the port microprocessor senses CCCBUSAVAIL and has data available for transfer to the CBus.

When the channel places the last word on the CBus during a device write operation, it asserts CBus LAST WORD. In response, the port CBus control logic asserts condition code CBus last word (CCCBLSTWD). When the port microprocessor detects CCCBLSTWD, it responds with a stop CBus (MPSTOPCBUS) command. This causes the port CBus control logic to assert CBus DONE during the next port data cycle. The port will make no more data requests during subsequent request cycles. CBus DONE causes the channel to terminate the operation. CBus READY is negated when the channel is prepared to begin another data transfer.

The port microprocessor can also execute a store CBus status information (MPSTORECBUS) command with an MPSTOPCBUS command on the same microcycle. This causes the port CBus control logic to assert CBus STORE along with CBus DONE on the next port data cycle. Asserting CBus STORE and CBus DONE during the same data cycle forces the channel to store channel status in the channel's assigned reset and status logout area.

NOTE
MPSTORECBUS should never be asserted unless MPSTOPCBUS is also asserted on the same microcycle.

The port microprocessor executes both the MPSTOPCBUS command and MPSTORECBUS command, causing the port CBus control logic to assert CBus DONE and CBus STORE, when it has transferred all data over the CBus during a device read operation. The port microprocessor also executes these commands when it detects, during a read or write, one of the following transfer error condition codes set:

CBus parity error (CCCBUSPARERR)
CMVR parity check (CCCMVRPARCHK)
CBus channel error (CCCHANERR)
PLI parity error (CCPLIPARERR).

The port will make no more data requests during subsequent request cycles.

5.2.14 PLI Input Buffer

The PLI input buffer is a latched 9-bit (8 data bits + 1 parity bit) register that passes data from the PLI bus to the mover/formatter. The register is loaded with a receive PLI MPRECVPLI command at CLK4 whenever an 8-bit byte is present for input from the PLI bus. Every time the register is loaded, PLI bus parity is loaded into a holding flip-flop.

Once the register is loaded, the port microprocessor transfers the data (four bits at a time) to the mover/formatter (four bits at a time) with the commands MPPLINTOFMTR, MPLFTNIBFIRST, MPSHIFTRIGHT, and so on. The port microprocessor can also execute an MPPLINTOCBUF command, transferring 8-bit data byte to the eight MBus LSBs, for further transfer to one of the microprocessor's internal storage media.

5.2.15 PLI Parity In Checker

The PLI Parity In checker checks for odd parity on every 8-bit data byte that the port reads from the PLI input buffer. If parity is incorrect, condition code PLI parity error (CCPLIPARERR) is generated and passed to the microsequencer's condition code multiplexer. The port microprocessor sets DATA PATH ERR (CSR26) in the CSR when it senses CCPLIPARERR set. CCPLIPARERR stays latched until cleared by the microprocessor.

5.2.16 PLI Output Buffer

The PLI output buffer is a latched 9-bit (8 data bits + 1 parity bit) register that passes data from the mover/formatter to the PLI bus. The port loads the register (via the PMUX) when it has an 8-bit byte assembled and ready for transfer to the PLI bus from either the eight mover/formatter MSBs (PLI output byte) or the eight MBus LSBs (via the CBUF). The port microprocessor loads the register with either a CBUF to PLI output buffer (MPCBUFTOPLOUT) command at CLK4 time, or a formatter to PLI output buffer (MPFMTRTOPLOUT) command at CLK2 time. When the register is loaded, odd parity is generated and loaded into a holding flip-flop.

After the register is loaded, the port enables the tristate register outputs with a transmit PLI (MPXMITPLI) command, placing the data on the PLI bus.

5.2.17 PLI Parity Out Generator

The PLI parity out generator normally generates odd parity for every 8-bit data byte passed from the port to the PLI output buffer. The parity bit is latched into a holding flip-flop for output to the PLI bus. The diagnostic command, test PLI parity generator (MPTESTPLIPAR), forces the PLI parity generator to generate even parity.

5.2.18 PLI Control Logic

The PLI control logic arbitrates the PLI protocol and the port microprocessor protocol for accessing the PLI, and performs synchronization functions between the PLI and the CMVR module. Figures 4-8 and 4-9 show the PLI signals, and Tables 4-7, 4-8, 4-9, and 4-10 describe the PLI signals.

5.2.19 Parity Predictor

Because the four mover/formatter registers may be serial or parallel loaded and read, and shifted in several different ways, correct parity cannot be simply propagated through the mover/formatter. Therefore, a parity predictor is used. The parity predictor enables the port microprocessor to verify data integrity through the mover/formatter using combined hardware and microcode functions to predict correct parity.

The parity predictor includes a J-K flip-flop, two 4-bit parity checkers, and related control logic. The J-K flip-flop is toggled every time a CBus or PLI parity bit is detected on a data transfer in either direction. The output of the J-K flip-flop, condition code mover parity check (CCMVRPARCHK), is monitored by the microprocessor.

The parity predictor is primarily controlled by several microprocessor commands that simultaneously control other functions, and therefore does not require much separate microcode. However, the parity predictor does require two unique commands for proper control:

1. Industry-Compatible Mode (MPINDSTCOMP) -- sets the parity predictor to operate correctly in industry-compatible mode (see Figure 5-10). It enables the parity predictor to calculate correct parity for CBus D32-D35, which do not pass through the mover/formatter in industry-compatible mode. The calculated parity for D32-D35 is then subtracted from the calculated parity for the entire 36-bit CBus word.

   This command is executed when transferring data from the CBus to the PLI in industry-compatible mode.

2. Clear parity check (MPCLRPARCHK) -- clears CCMVRPARCHK.

To detect an error, different microcode algorithms are used to predict the state of CCMVRPARCHK that should correspond to the number of parity bits detected at the CBus and PLI interfaces during data transfers. If the state of CCMVRPARCHK is not correct when it is checked by the microcode, then it is likely that an error has occurred in the mover/formatter. When the port microproccessor senses the incorrect state of CCMVRPARCHK, it sets DATA PATH ERR (CSR26) in the CSR.

There are six microcode algorithms, each slightly different according to data format mode (see Figures 5-9 through 5-11) and direction of transfer. The algorithms check the following conditions:

1. HIGH DENSITY -- CBus to PLI -- CCMVRPARCHK is always toggled an odd number of times for every two-word (9-byte) transfer from the CBus to the PLI.

2. HIGH DENSITY -- PLI to CBus -- CCMVRPARCHK is always toggled an odd number of times for every two-word (9-byte) transfer from the PLI to the CBus.

3. INDUSTRY COMPATIBLE -- CBus to PLI - CCMVRPARCHK is always toggled an even number of times for every one-word (4-byte) transfer from the CBus to the PLI. The port microprocessor executes an MPINDSTCOMP command for every transfer.

4. INDUSTRY COMPATIBLE -- PLI to CBus -- CCMVRPARCHK is always toggled an even number of times for every one-word transfer (4-byte) from the PLI to the CBus.

5. CORE DUMP -- CBus to PLI -- CCMVRPARCHK is always toggled an even number of times for every two-word (10-byte) transfer from the CBus to the PLI.

6. CORE DUMP -- PLI to CBus -- CCMVRPARCHK is always toggled an even number of times for every two-word (10-byte) transfer from the PLI to the CBus.

5.3 PORT MICROPROCESSOR

The port microprocessor comprises the following, all of which, except the microprocessor ALU, are located on the microprocessor control module.

The Am2901-based microprocessor ALU (physically located on the EBus interface/port ALU module, and described in Section 5.1.

- The microprocessor controller, which includes

  An Am2910 microsequencer and associated control, input, and output functions

  The 4K-word by 60-bit control RAM (CRAM), including a load and read/verify path to the MBus

  The 60-bit CRAM control register, which latches the currently executing microword.

- The 1K-word by 36-bit local RAM storage memory that interfaces to the MBus and is read and written by the microprocessor

- Microprocessor control logic to control various port microprocessor timing functions.

5.3.1 Condition Code Multiplexer

The condition code multiplexer (CCMUX) is a 16-input by 1-bit multiplexer that enables one of 16 condition codes to alter the microword execution sequence.

Using microword skip/condition (MWSKIPFLD01-04), bits 44-47, the port microprocessor selects the CCMUX input to pass to the Am2910 TEST COND input. In the same microcycle, the port microprocessor enables the test condition by asserting MWCCENA (described below) on the Am2910 TEST EN input. The state of the selected condition code affects the conditional microsequencer instructions -- such as certain jump, return, load instuctions. The result of these instructions determines the address of the next microword to be executed and the sequence of microprogram execution.

The source of the conditon codes is the CMVR module (00, 05-07, and 13-17), the microprocessor ALU (02 and 12), or the EBus control logic (01, 03-04, and 10-11). The 16 condition codes are described in Table 5-7.

Table 5-7 Condition Code Definitions

Condition Code/ MWSKIPFLD 01-04	Definition
CCCBUSAVAIL 00	CBus available -- asserted when the CBus input buffer is available to receive a word from the CBus, or the CBus output buffer is available to be loaded with a word for transfer to the CBus, or the CBus is not currently active (no data transfers occurring).
CCGRNTCSR 01	Grant CSR -- enables port microprocessor access to the control and status register. Asserted by microprocessor request CSR (MPRQSTCSR) if the KL10 does not have access to the CSR.
CCFEQ0 02	ALU function = 0 -- indicates that the result of the last ALU operation was all zeros.
CCCSRCHNG 03	CSR register changed -- asserted when the KL10 writes the CSR with a CONO, or an EBus parity error is detected. Cleared when the port microprocessor asserts read CSR (MPREADCSR). The condition code notifies the port microprocessor that the KL10 has changed the contents of the CSR.
CCEBPARERR 04	EBus parity error -- asserted when an EBus parity error is detected.
CCRCVRBUFAFUL 05	Receiver buffer A full -- originates in the PLI. Asserted when receive buffer A in the packet buffer module is loaded with a CI packet.
CCRCVRBUFBFUL 06	Receiver buffer B full -- originates in the PLI. Asserted when receive buffer B in the packet buffer module is loaded with a CI packet.
CCXMTRATTN 07	Transmitter attention -- originates in the PLI. Asserted when the transmit buffer in the packet buffer module requires attention.
CCEBUSRQST 10	EBus request -- asserted by an EBus DATAO or DATAI when the port is in the MPROC RUN state (CSR32 set).

Table 5-7 Condition Code Definitions (Cont)

Condition Code/ MWSKIPFLD 01-04	Definition
CCINTRACTIVE 11	Interrupt active -- Indicates that the PI level 01 through 07 interrupt request, previously executed by the port microprocessor, is waiting for KL10 processing.
CCMBSIGN 12	ALU sign bit set -- indicates that the last ALU operation set the sign bit (MSB or bit 00).
CCMVRPARCHK 13	Mover parity check -- toggled when a parity bit = 1 is sensed from either the CBus or the PLI, during DMA data transfers between them. By comparing the value (1 or 0) of this condition code with a predicted value, the microprocessor determines if a parity error occurred during a data transfer through the mover/formatter.
CCCBUSPARERR 14	CBus parity error -- asserted when a parity error is detected in a word read from the CBus. The condition code is latched until it is cleared by a command from the microprocessor controller.
CCPLIPARERR 15	PLI parity error -- asserted when a parity error is detected in a word read from the PLI. The condition code is latched until it is cleared by a command from the microprocessor controller.
CCCHANERR 16	CBus channel error -- asserted when the CBus ERROR signal is asserted on the CBus. The condition code is latched until it is cleared by a command from the microprocessor controller.
CCCBLSTWD 17	CBus last word -- asserted when the CBus LAST WORD signal is asserted on the CBus. The condition code is latched until it is cleared by a command from the microprocessor controller.

The field microword condition code enable (MWCCENA), bit 33, controls the way the microsequencer tests the condition codes. If MWCCENA is not asserted, then the condition is always met. For example, a conditional jump instruction will always branch rather than execute the next sequential microword, thus behaving like an

unconditional jump. If MWCCENA is asserted, the condition is met only if the condition code selected through the CCMUX by MWSKIPFLD01-04 is also asserted. For example, a conditional jump instruction will branch if the selected condition code is asserted. If the selected condition code is not asserted, then the next sequential microword is executed.

5.3.2 Microsequencer

The microsequencer is an Am2910 microprocessor sequence controller (see Figure 5-14). It selects the address of the next microword to be executed. The Am2910 is configured such that:

The outputs (NXTADDR11-00) are always enabled, by tying OUT EN to ground.

The microprogram counter (μPC) is always incremented on the next clock by tying CIN to the incrementer (+1) to +3 V.

The REGISTER/COUNTER force load feature is disabled by tying LOAD REG/CTR to +3 V.

The PL, MAP, and VECT outputs are not connected to enable other external sources for the DIRECT INPUTS.

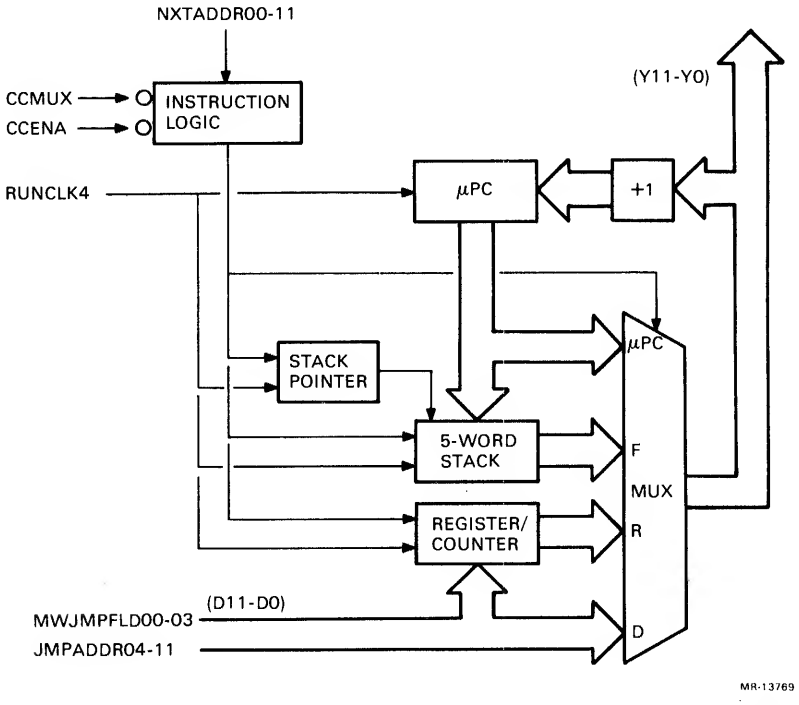


Figure 5-14 Am2910 Block Diagram (Simplified)

The microsequencer is controlled by 16 instructions. The instruction operation code is encoded in microword 2910 control, (MWCTRLFLD00-03) bits 52-55), the I0-I3 inputs to the microsequencer. The instructions are described in Table 5-8.

Table 5-8 Microsequencer Instructions

Opcode/ Microcode Mnemonic	Description
0 JMPZ	Jump to address zero -- not used.
1 CJSR	Conditional jump to subroutine -- if both MWCCENA and the condition code selected by the MWSKIPFLD are asserted, transfer the address on the D-inputs to the Y-outputs, and push the contents of the μPC (that is, Y+1) on the stack. Otherwise, transfer the contents of μPC to the Y-outputs.
2 JMAP	Jump using MAP output -- not used.
3 CJMP	Conditional jump -- if both MWCCENA and the condition code selected by the MWSKIPFLD are asserted, transfer the address on the D-inputs to the Y-outputs; otherwise, transfer the contents of the μPC to the Y-outputs.
4 PUSH	Push and conditionally load counter -- push the contents of the μPC on the stack. If both MWCCENA and the condition code selected by the MWSKIPFLD are asserted, load the register/counter with the value on the D-inputs.
5 CJSR.RP	CJSR using pipeline or counter -- not used.
6 CONVEC	Conditional vector jump -- not used.
7 CJ.RP	CJMP to counter or pipeline address -- if both MWCCENA and the condition code selected by the MWSKIPFLD are asserted, transfer the address on the D-inputs to the Y-outputs; otherwise, transfer the address held in the register/counter to the Y-outputs.
10 LOOP.ON.CNT	Repeat loop if counter not zero -- if the contents of the register/counter are not zero, decrement the register/counter and transfer the address from the top of the stack to the Y-outputs. Decrement the stack pointer (POP); Otherwise, transfer the contents of μPC to the Y-outputs.

Table 5-8 Microsequencer Instructions (Cont)

Opcode/ Microcode Mnemonic	Description
11 REPEAT	Repeat instruction if counter not zero -- if the contents of the register/counter are not zero, decrement the register/counter and transfer the address from the D-inputs to the Y-outputs. Otherwise, transfer the contents of μPC to the Y-outputs.
12 CONRET	Conditional subroutine return -- if both MWCCENA and the condition code selected by the MWSKIPFLD are asserted, transfer the address from the top of the stack to the Y-outputs and decrement the stack pointer; otherwise, transfer the contents of ~PC to the Y-outputs.
13 CJ.PL	CJMP using pipeline and POP -- if both MWCCENA and the condition code selected by the MWSKIPFLD are asserted, transfer the address on the D-inputs to the Y-outputs and decrement the stack pointer; otherwise, transfer the contents of the ~PC to the Y-outputs.
14 LOADCNT	Load counter -- load the value/address on the D-inputs into the register/counter.
15 TEST.LOOP	Test end-of-loop condition -- not used.
16 CONT	Continue normally -- not used.
17 3WAYBRANCH	Three-way branch -- not used.

The first microword that the port microprocessor excutes on initial startup (the KL10 set MPROC RUN, CSR32) is an unconditional jump (that is, a CJMP with MWCCENA not asserted). This guarantees that the Am2910 register/counter is correctly loaded on the first executed instruction.

5.3.3 RAM Address Register

The RAR is a 13-bit register that addresses the next CRAM location to write or read. The RAR is loaded from MBus D01-D13 when the KL10 executes a DATOLOADRAR (DATAO with EBus D00 = 1). It is used to load and read/verify the contents of the CRAM when the port is not running (CSR32 reset). The RAR also holds the starting CRAM address. When the port microprocessor is initially started (CSR32 set), the first CRAM location is always addressed by the RAR rather than by the microsequencer.

The 60-bit CRAM word is written and read/verified over the 36-bit MBus. Therefore, it is written and read in two 30-bit half-words, selected by the RAR LSB as follows:

RAR12 = 0 Select the right CRAM bank (least significant half-word, CRAM30-59).

RAR12 = 1 Select the left CRAM bank (most significant half-word, CRAM00-29).

Because the register is not also an up/down counter, it is loaded every time the KL10 wants to address a CRAM location. In order to write or read a CRAM location, the KL10 must execute four commands: a DATOLOADRAR to address the first CRAM half-word, followed by either a DATOLOADMW or DATIREADMW to write or read the CRAM half-word; and then another DATOLOADRAR and a DATOLOADMW or DATIREADMW to write or read the other half-word.

When the port is being operated in single-cycle mode (CSR22 set), the KL10 loads the next address to be executed into the RAR at the end of each single cycle. The next address is contained in the latch address register (LAR), (see section 5.3.4).

The KL10 can load only the RAR. It can read only the contents of the RAR through the LAR.

5.3.4 Latch Address Register

The LAR is a 13-bit register that latches the CRAM address on every microcycle. It is a diagnostic tool. The KL10 reads the LAR by setting DIAG SEL LAR (CSR21) with a CONOLOADCSR, and executing a DATAI when the port is not running (CSR32 reset).

If the port microprocessor halts while in the MPROC RUN state (CSR32 set), the LAR contains the address of either the last CRAM location executed or the next CRAM location to be executed. The LAR contents are determined by the state of DIAG SINGLE CYCLE (CSR22) as follows:

If the port is not running in the single-cycle state (DIAG SINGLE CYCLE, CSR22 is not set) and the port microprocessor halts for any reason, the LAR contains the address of the last microword executed.

If the port is running in the single-cycle state (DIAG SINGLE CYCLE, CSR22 set), it automatically halts at the completion of each microcycle. The LAR contains the address of the next microword to be executed. The KL10 executes a DATIREADLAR to get the address, a DATOLOADRAR to load the address back into the RAR, and a CONOLOADCSR to set MPROC RUN (CSR32) and enable the port microprocessor to execute the next single-cycle when it is restarted.

Reading the LAR destroys the current E buffer data. Therefore, to preserve valid E buffer data, the KL10 must read the E buffer (execute a TESTREADEBUF DATAI) before it reads the LAR. After it reads the LAR, the KL10 must execute a TESTLOADEBUF DATAO to restore the preserved data to the E buffer. The 12 LAR MSBs are loaded by CRAMADDR00-11 from the ADDR MUX (see Section 5.3.5). The LSB is loaded from RAR12. The LAR is loaded at CLK1 time of every microcycle when the port is not in the single-cycle state, or CLK 4 time of every microcycle when the port is in the single-cycle state. The LAR output is MBus D01-13. MBus D14-35 are undefined during an LAR read.

5.3.5 Address Multiplexer

The address multiplexer (ADDR MUX) is a two-input by 12-bit multiplexer that passes either the Am2910 microsequencer Y-outputs (NXTADDR00-11) or RAR00-11 to the CRAM address inputs (see Figure 5-15).

In the MPROC RUN state (CSR32 set) the next address is normally fetched from the Am2910 Y-outputs. But, when CSR32 is initially set, the address of the first CRAM location to be executed is always fetched from the RAR.

When the port is not running (CSR32 reset) the CRAM address is always fetched from RAR.

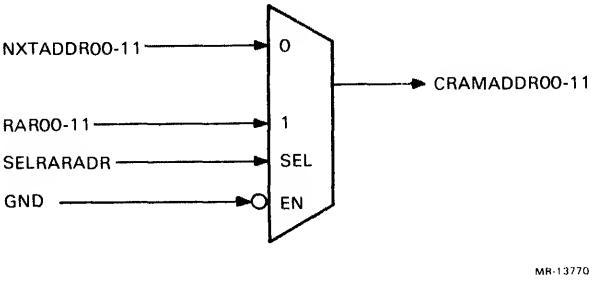


Figure 5-15 Address Multiplexer (Simplified)

Note that RAR12 is not passed through the multiplexer, but goes directly to the microprocessor control logic to select the CRAM half-word.

5.3.6 Control RAM

The CRAM is a 4K-word by 60-bit RAM with tristate input/output and 55 ns access time. It stores the port microprocessor microcode.

The CRAM is initially loaded and undergoes read/verify, one half-word at a time, from the 30 MBus LSBs (MBus D06-D35) when the port is not running (MPROC RUN, CSR32 not set). To load one CRAM location, four EBus transfers are needed, in the following sequence:

1. DATOLOADRAR -- load the RAR with the address of the right bank CRAM location. The KL10 executes a DATAO with EBus D00 = 1, D01-D12 = address, D13 = 0 (least significant, or right half). EBus bits D14-D35 are undefined.

2. DATOLOADMW -- load data into the right half of the CRAM location. The KL10 executes a DATAO with EBus D00 = 0, and D06-D36 = data. EBus bits D00-D05 are undefined.

3. DATOLOADRAR - load the RAR with the address of the left bank CRAM location. The KL10 executes a DATAO with EBus D00 = 1, D01-D12 = address, D13 = 1 (most significant, or left half). EBus bits D14-D35 are undefined.

4. DATOLOADMW - load data into the left half of the CRAM location. The KL10 executes a DATAO with EBus D00 = 0, and D06-D36 = data. EBus bits D00-D05 are undefined.

When the port microprocessor is running (MPROC RUN, CSR32 set), the CRAM location currently addressed by the microsequencer is normally strobed into the CRAM register (see Section 5.3.9) by RUNCLK1 of every microcycle, for execution. However, when DIAG SINGLE CYCLE (CSR22) is set or during initial microprocessor startup, the address of the first CRAM location is always taken from the RAR instead of the Am2910.

If either a CRAM parity error or an MBUS error occurs while the microprocessor is running, CRAM data may be invalid. The entire CRAM should be reloaded before the port microprocessor is restarted.

5.3.7 CRAM Load Buffers

The two CRAM load buffers, left CRAM load buff and right CRAM load buff, are used to load the CRAM when the port is not running (CSR32 reset). Each is a 30-bit tristate buffer that inputs EBus data, via the MBus, to the CRAM I/O pins (see Figure 5-16).

The CRAM is loaded when the KL10 executes a DATAOLOADMW (a DATAO with EBus D00 = 0 and the state of EBus D01-D05 is undefined). This DATAO also causes EMUX EN, CRAM WREN, and (with RAR12) the load buffer enables to be asserted. The complete CRAM load sequence is described in Section 5.3.6.

The CRAM load buffers are pass-through buffers and do not latch the data. The left CRAM load buff passes MBus D06-D35 to CRAM bits 00-29, and the right CRAM load buff passes MBus D06-D35 to CRAM bits 30-59. The buffers are enabled only when the port is not running (CSR32 reset).

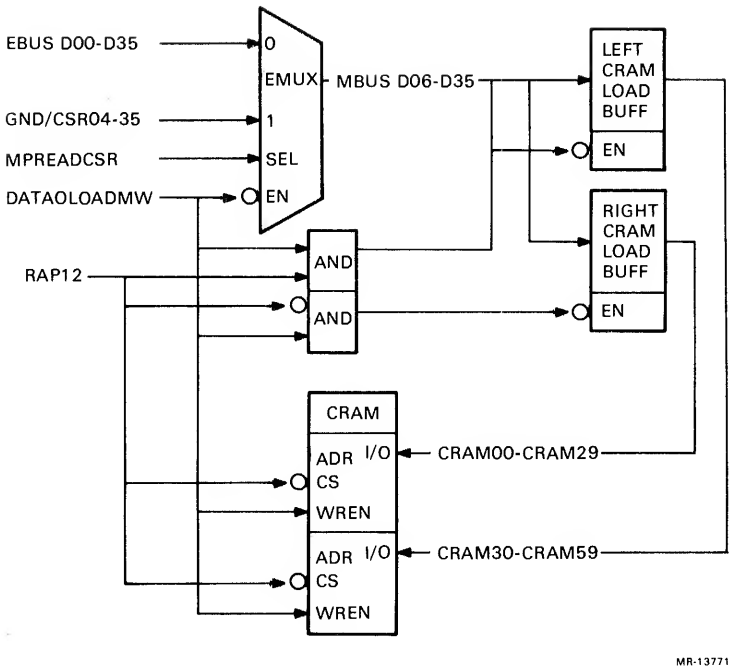


Figure 5-16 CRAM Load Buffers (Simplified)

5.3.8 CRAM Parity Checker

The CRAM parity checker checks for odd parity on the 59 MSBs of the 60-bit microword in the CRAM register. The LSB, MWMARKBIT bit 59, is not included in the parity check. If parity is incorrect (even), CRAM PARITY ERR (CSR06) and RQST INTERRUPT (CSR05) are set in the CSR, and the port microprocessor is halted. A nonvectored (40 + 2n) interrupt request will be generated over the EBus.

Microword parity is calculated and the state of MWPAR (bit 12) is set by the microcode assembler to give the microword odd parity.

CRAM PARITY ERR (CSR06) can be force-set in order to halt the port microprocessor at a specific location (breakpoint). It is then cleared by the KL10 executing a CONO with EBus D24 = 1 (EBUS PARITY ERR, CSR24). The port microprocessor is restarted by setting MPROC RUN (CSR32).

5.3.9 CRAM Register

The CRAM register is a 60-bit register that holds the currently executing microword. It is loaded from the currently addressed CRAM location by RUNCLK1 of every microcycle. Then RUNCLK2, RUNCLK3, and RUNCLK4 clock the execution of the operations specified by the microword fields.

RUNCLK1, RUNCLK2, RUNCLK3, and RUNCLK4 are gated outputs of CLK1, CLK2, CLK3, and CLK4 respectively. They are active only when the microprocessor is in the MPROC RUN state (CSR32 set).

5.3.10 Microword Field Definitions

The definitions of the microword fields are described in Table 5-9.

Table 5-9 Microword Field Definitions

Field/ Bits	Definition
MWJMPFLD 00-11	CRAM 00-11, jump field -- one source for all or part of the next CRAM location address. MWJMPFLD bits 00-03 are input to Am2910 D11-D8. MWJMPFLD bits 04-11 or MBus D16-D23 are input to Am2910 D7-D0 via the JMP MUX.
MWPAR	CRAM 12, parity bit) -- microword parity bit. Its state is calculated and set by the microcode assembler to give the word odd parity. Even microword parity will generate CRAMPE (CRAM parity error).
MWOUTPUTENA	CRAM 13, ALU data output enable -- controls the Am2901 OUT EN. When asserted, this bit enables the Am2901 tristate Y-outputs to be asserted on the MBus.
MWMGCFLD 00-09	CRAM 14-23, magic number field -- provides constants for the Am2901 internal RAM, provides local RAM storage memory addresses, and with other microword fields, controls the EBus and CMVR interfaces (see MWBUSCTLFLD, MWSKIPFLD, and MWRAMODE).
MWSORCEFLD 00-02	CRAM 24-26, ALU source field I2-I0 -- selects the source of the R and S inputs to the Am2901 ALU, as follows: MWSORCEFLD 00-02 R S 0 A Q 1 A B 2 Z Q 3 Z B 4 Z A 5 D A 6 D Q 7 D Z Where: A = The contents of the Am2901 internal RAM location addressed by MWPORTAFLD 00-03 B = The contents of the Am2901 internal RAM location addressed by MWPORTBFLD 00-03

Table 5-9 Microword Field Definitions (Cont)

Field/ Bits	Definition
	D = The data on CNST 00-09, MBus 10-25, and CONST 26-35 Q = The contents of the Am2901 internal Q register Z = Zero.
MWFUNCTFLD 00-02	CRAM 27-29, ALU function I5-I3 -- controls the functions that the Am2901 ALU performs on the R and S inputs.

MWFUNCTFLD 00-02	Function
0	R plus S
1	S minus R
2	R minus S
3	R or S
4	R and S
5	-R and S
6	R xor S
7	R xnor S

MWDESTFLD 00-02

CRAM 30-32, ALU destination I8-I6 -- determines if the Am2901 output (Y) will be from the ALU or the internal RAM location addressed by MWPORTAFLD 0-3. Also determines the input to the internal Q register and to the internal RAM location addressed by MWPORTBFLD 0-3.

This field also controls the internal RAM shifter and Q shifter, causing the RAM and Q register inputs to multiplied or divided by 2 (shifted left or right). Therefore, by controlling the Am2901 internal data paths, this field determines the destinations of the Am2901 internal data. The effect of this field is as follows:

MWDESTFLD 00-02	Y Gets	Q Gets	RAM B Gets
0	ALU	ALU	HOLD
1	ALU	HOLD	HOLD
2	RAM A	HOLD	ALU
3	ALU	HOLD	ALU
4	ALU	Q/2	ALU/2
5	ALU	HOLD	ALU/2
6	ALU	QX2	ALUX2
7	ALU	HOLD	ALUX2

Table 5-9 Microword Field Definitions (Cont)

Field/ Bits	Definition
MWCCENA	CRAM 33, condition code enable bit -- the Am2910 microsequencer condition code enable bit: MWCCENA not asserted = conditional test always passed; MWCCENA asserted = conditional test passed only if CCMUX input is asserted.
MWRAMODE	CRAM 34, local storage RAM mode bit -- selects either local or global addressing to address the local storage RAM: MWRAMODE not asserted = global addressing; MWMGCFLD 00-09 contains the entire address of a location in the local storage RAM. MWRAMODE asserted = local addressing. MWMGCFLD 05-09 contains the five address LSBs, and SADREG 00-04 contains the five address MSBs of a location in the local storage RAM.
MWPORTAFLD 00-03	CRAM 35-38, port A address field A3-A0 -- the Am2901 internal RAM port A address field, which addresses one of 16 RAM locations that will be read through the A latch. This port is read-only.
MWPORTBFLD 00-03	CRAM 39-42, port B address field B3-B0 -- The Am2901 internal RAM Port B address field. This field addresses one of 16 RAM locations that will be either read through the B latch or written. The Port A and B locations are:

A/B 00-03	Name	Description
00	T0	Temporary register 0
01	T1	Temporary register 1
02	T2	Temporary register 2
03	T3	Temporary register 3
04	T4	Temporary register 4
05	LNGTH	The length of something
06	CMD	Command or message to process
07	FLAG/ FLAGS	Latest state flags (global flag word)
10	REG	
11	INTLK	Address of interlock word for the command queue being processed
12	FLINK	Forward link of a queue
13	BLINK	Backward link of a queue
14	OFFSET	Base local storage address of command queue being processed
15	SPARE	

Table 5-9 Microword Field Definitions (Cont)

Field/ Bits	Definition		
	16	R.MASK	Mask to isolate right half of word
	17	L.MASK	Mask to isolate left half of word

MWSKIPFLD 00-04

CRAM 43-47, skip field -- this field is decoded by the microprocessor COND/SKIP decoder and the condition code multiplexer (CCMUX).

The CCMUX decodes only MWSKIPFLD01-04, and ignores MWSKIPFLD00. The decoded function selects one of 16 condition code inputs to the CCMUX, for input to the Am2910, as follows:

MWSKIPFLD 01-04	CCMUX INPUT
00	CCCBUSAVAIL
01	CCGRNTCSR
02	CCFEQ0
03	CCCSRCHNG
04	CCEBPARERR
05	CCRCVRBUFAFUL
06	CCRCVRBUFBFUL
07	CCXMTRATTN
10	CCEBUSRQST
11	CCINTRACTIVE
12	CCMBSIGN
13	CCMVRPARCHK
14	CCCBUSPARERR
15	CCPLIPARERR
16	CCCHANERR
17	CCCBLSTWD

The COND/SKIP decoder decodes only MWSKIPFLD00, 02-04, and ignores MWSKIPFLD01. The decode functions are:

MWSKIPFLD 00, 02-04	Function
20	LOADSADREG -- causes the local storage address register to be loaded with the contents of MWMGCFLD05-09
21	SELMBUSFLD -- selects MBus D16-D23 through the jump multiplexer as input to Am2910 D7-D0 (MWJMPFLD00-03 are input to Am2910 D11-D8)

Table 5-9 Microword Field Definitions (Cont)

Field/ Bits	Definition	
	22	RDLOCALMEM -- causes the contents of the currently addressed local storage RAM location to be placed on the MBus
	23	LDLOCALMEM -- causes the currently addressed local storage RAM location to be loaded from the MBus
	24	SELCNSTFLD -- selects MWMGCFLD00-09 through the constant multiplexer as the 10 least significant and the 10 most significant D inputs to the Am2901 ALU.
MWBUSCTLFLD 00-02	CRAM 48-50, bus control field -- with the MWMGCFLD field, controls the various functions of the EBus and the CMVR interfaces. The field is decoded as follows:	
	MWBUSCTLFLD 00-02	**Function**
	0	No function
	1	Select PLI
		MWMGCFLD
		00-01 No function
		02-05 PLI LINK CONTROL 0-3 -- passes PLI LINK CONTROL 0-3 to the PLI bus
		06 MPSELECTPLI (select PLI) -- asserts the PLI SELECT line
		07 MPXMITPLI (transmit PLI) -- enables the PLI output buffer tristate outputs to the PLI bus
		08 MPRECVPLI (receive PLI) -- loads the contents of the PLI Bus into the PLI input buffer
		09 MPTESTPLIPAR (test PLI parity) -- causes the PLI PAR OUT generator to generate even (bad) parity
	2	Select MBus

Table 5-9 Microword Field Definitions (Cont)

Field/ Bits	Definition	
		MWMGCFLD
		00-01 No function
		02 MPENACMUX -- enable CMUX. Enables the CMUX tri-state outputs to the MBus
		03 MPPLINTOCMUX -- PLI into CMUX. Enables the PLI input path to the CMUX, allowing the PLI input buffer to be asserted on the 8 MBus LSBs
		04 MPCBUFTOPLOUT -- CBUF to PLI output buffer. Loads the 8 C buffer LSBs into the PLI output buffer
		05 MPCLRCCCODE -- clear condition code. Causes all of the condition code status bits (except CCMVRPARCHK) on the CMVR module to be cleared
		06 MPCLRPARCHK -- clear parity check. Causes the condition code CCMVRPARCHK to be cleared
		07-09 No function
	3	Select FMTR
		MWMGCFLD
		00-01 No function
		02 MPSHFTFMTR8 -- shift formatter 8 bits. Causes the contents of the mover/formatter to be shifted eight bits to the left or right, depending on the state of the command MPSHIFTRIGHT
		03 MPSHFTFMTR4A -- shift formatter 4 bits. Causes the contents of the mover/formatter to be shifted four bits to the left or right, depending on the state of the command MPSHIFTRIGHT

Table 5-9 Microword Field Definitions (Cont)

Field/ Bits	Definition	
	04	MPCBUFTOFMTR -- CBUF to formatter. Causes data previously stored in the C buffer to be loaded into the mover/formatter register
	05	MPPLINTOFMTR -- PLI input buffer to formatter. Causes the 8-bit data byte currently stored in the PLI input buffer to be shifted (four bits at a time) into the mover/formatter serial input lines. This command is executed with MPSHFTFMTR4A or MPSHFTFMTR8. If executed with MPSHFTFMTR4A, then only four PLI input buffer bits are shifted into the mover/formatter. If executed with MPSHFTFMTR8, all 8 PLI input buffer bits are shifted in. The state of the MPRHTNIBFIRST command determines if the four LSBs or MSBs are shifted in first. If MPSHIFTRIGHT is asserted, the 4-bit nibbles are shifted into the four mover/formatter MSBs and shifted right. Otherwise, they are shifted into the four mover/formatter LSBs and shifted left
	06	MPFMTRTOPLOUT -- formatter to PLI output buffer. Causes the 8-bit data byte currently stored in the mover/formatter PLI output register to be loaded into the PLI output buffer
	07	MPSHIFTRIGHT -- shift right. When asserted, either the currently selected PLI input buffer nibble or the four mover/formatter LSBs (MVROUT36-39) are shifted into the four mover/formatter MSB serial input lines and shifted right. If MPPLINTOFMTR is asserted, then the currently

Table 5-9 Microword Field Definitions (Cont)

Field/ Bits	Definition	
		selected PLI input buffer nibble is shifted into mover/formatter MSBs and shifted right. If MPPLINTOFMTR is not asserted, then MVROUT36-39 are shifted into the mover/formatter MSBS and shifted right. Two 4-bit shifts can be executed in one microcycle, right-shifting an 8-bit byte in one microcycle
	08	MPRHTNIBFIRST -- right nibble first. When asserted, the four PLI input buffer LSBs are shifted into the mover/formatter serial input lines first. When not asserted, the four PLI input buffer MSBs are shifted into the mover/formatter serial input lines first
	09	MPZEROLFTNIB -- zero left nibble. When asserted, causes the four mover/formatter PLI output register MSBs to be forced to zero before they are loaded into the PLI output buffer
4	Select CBus	
	MWMGCFLD	
	00-01	No function
	02	MPSTARTCBUS -- start CBus. Causes CBus START and CBus RESET to be asserted on the CBus at the proper time during the next CBus SELECT cycle
	03	MPSTOPCBUS -- stop CBus. Causes CBus DONE to be asserted on the CBus at the proper time during the next CBus SELECT cycle
	04	MPSTORECBUS -- store CBus. Causes CBus STORE to be asserted on the CBus at the

Table 5-9 Microword Field Definitions (Cont)

Field/ Bits	Definition	
		proper time during the next CBus SELECT cycle. This command is executed at the same time as MPSTOPCBUS
	05	MPWRITEMEM -- write memory. Causes CBus CTOM to be asserted on the CBus at the proper time during the next CBus SELECT cycle. This command is executed at the same time as MPSTARTCBUS for a data transfer to KL10 memory
	06	MPINDSTCOMP -- industry compatible. Enables the parity predictor to predict correct parity in industry-compatible mode. This command is executed in industry-compatible mode when transferring data from the CBus to the PLI interface.
	07	MPCBINTOFMTR -- CBus input buffer to formatter. Causes the contents of the CBus input buffer to be loaded into the mover/formatter
	08	MPFMTRTOCBOUT -- formatter to CBus output buffer. Causes the contents of the mover/formatter to be loaded into the CBus output buffer
	09	MPSHFTFMTR4B -- causes the contents of the mover/formatter to be shifted four bits to the left
5	Select EBus	
	MWMGCFLD	
	00-01	No function
	02	MPLOADCSR -- load CSR. Loads EBUF00-EBUF17 into CSR00-CSR17
	03	MPREADCSR -- read CSR. Places CSR00-CSR35 on the MBus

Table 5-9 Microword Field Definitions (Cont)

Field/ Bits	Definition
04	MPRQSTCSR -- request CSR. Requests access to the CSR. Access is granted to the port microprocessor if the KL10 is not currently accessing the CSR.
05	MPLOADEBUS -- load EBus. Causes the contents of the E buffer to be asserted on the EBus
06	MPREADEBUS -- read EBus. Causes the contents of the EBus to be asserted on the MBus
07	MPLOADEBUF -- load EBUF. Causes port microprocessor data on the MBus to be loaded into the E buffer
08	MPRQSTINTR -- request interrupt. Causes the port to request an EBus PI level 01-07 interrupt (function 00-03). The interrupt function is determined by an IOP function control word, previously built and loaded into the E buffer by the port microprocessor. The microprocessor passes the IOP word to the EBus by asserting condition code EBus request (CCEBUSRQST)
09	MPEXORDEP -- examine or deposit. Causes the port to request an EBus examine or deposit PI Level 00 interrupt (function 04-07). The interrupt function is determined by an IOP function control word, previously built and loaded into the E buffer by the port microprocessor. The microprocessor passes the IOP word to the EBus by asserting Condition Code EBus Request (CCEBUSRQST)

Table 5-9 Microword Field Definitions (Cont)

Field/ Bits	Definition
	6 No function
	7 No function
MWCARRY	CRAM 51, carry input to Am2901 ALU -- this bit is the carry into the least significant bit of the ALU. When MWCARRY is 0, a zero is carried into the ALU LSB. When MWCARRY is 1, a one is carried into the ALU LSB.
MWCTRLFLD 00-03	CRAM 52-55, microsequencer control input field I0-I3 -- the instruction input field to the Am2910. Its functions are:

MWCTRLFLD 00-03	Instruction Mnemonic
0	JMPZ
1	CJSR
2	JMAP
3	CJMP
4	PUSH
5	CJSR.RP
6	CONVEC
7	CJ.RP
10	LOOP.ON.CNT
11	REPEAT
12	CONRET
13	CJ.PL
14	LOADCNT
15	TEST.LOOP
16	CONT
17	3WAYBRANCH

MWTIMEFLD	CRAM 56, time field -- when this bit is asserted, the current microinstruction execution time (microcycle) is extended from 320 ns to 480 ns.
MWSPARE00	CRAM 57, spare bit 00 -- no function
MWSPARE01	CRAM 58, spare bit 01 -- no function
MWMARKBIT	CRAM 59, mark bit -- has no microcode function. It is used only for hardware/microcode debug. The bit can be set in any microword, as an oscilloscope sync point. The bit is neither part of the microword parity calculation nor included in the CRAM parity check. Therefore, it can be set and cleared with no effect on the remainder of the microword content.

5.3.11 Microword Output Multiplexer

The microword output multiplexer (MW OUT MUX) is a two-input by 30-bit multiplexer (see Figure 5-17). It passes either the right or left half-microword (from the CRAM location currently addressed by the RAR) to MBus D06-D35. The six MBus MSBs are undefined. The multiplexer is enabled by READCRAM, which is asserted only by a DATIREADMW command from the KL10.

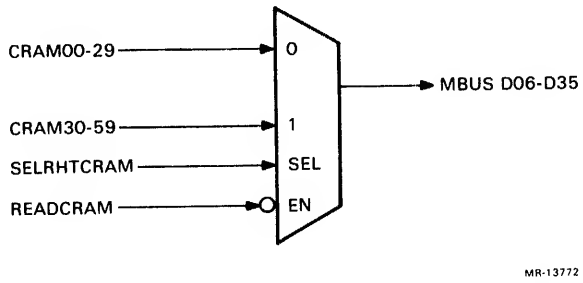


Figure 5-17 Microword Output Multiplexer (Simplified)

SELRHTCRAM (generated by RAR12) selects the half-microword input to the multiplexer as follows:

RAR12 = 0 Select the right CRAM bank (least significant half-word, CRAM30-59)

RAR12 = 1 Select the left CRAM bank (most significant half-word, CRAM00-29).

5.3.12 Jump Multiplexer

The jump multiplexer (JMP MUX) is a two-input by 8-bit multiplexer that passes either MWJMPFLD04-11 (jump field 04-11) or MBus D16-D23 to Am2910 D7-D0 (see Figure 5-18). MWJMPFLD00-03 are always input to Am2910 D11-D8.

MBus D16-D23 are selected as input to the jump multiplexer when MWSKIPFLD00, 02-04 = 21. (MWSKIPFLD01 is ignored.)

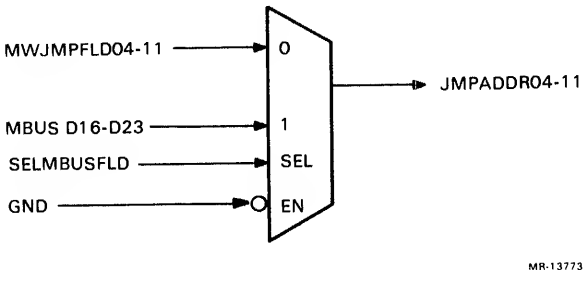


Figure 5-18 Jump Multiplexer (Simplified)

5.3.13 Local Storage RAM

The local storage RAM is a 1K-word by 36-bit tristate RAM with 55 ns access time. Its I/O pins are connected to the MBus. Approximately one-half of the local storage RAM locations are predefined for specific port functions, and many are completely or partially loaded at initialization.

The local storage RAM is addressed with either global or local addresses through the RAM mode multiplexer (see Section 5.3.14). It is loaded from the MBus at CLK2 time when MWSKIPFLD00, 02-04 = 23. Data is output from the local storage RAM to the MBus when MWSKIPFLD00, 02-04 = 22.

5.3.14 RAM Mode Multiplexer

The RAM mode multiplexer is a two-input 5-bit multiplexer that passes either MWMGCFLD00-04 or the contents of the local storage address register (STORADDR00-04) as the five MSBs of the local storage RAM address (ADR 0-4) MSBs. The microword MWRAMODE bit (bit 34) selects the input (see Figure 5-19).

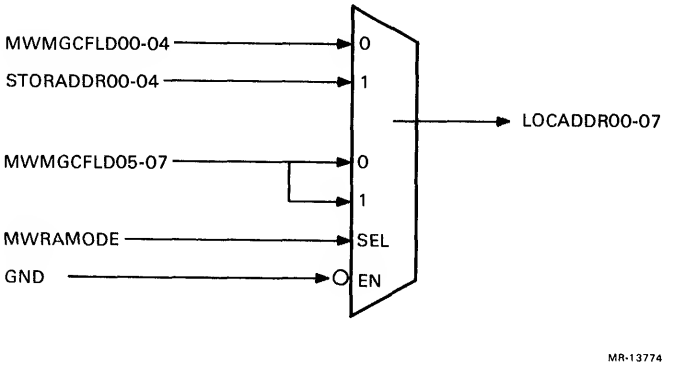


Figure 5-19 RAM Mode Multiplexer (Simplified)

The local storage RAM is addressed in one of two modes: global or local. In global-addressing mode, all 1024 RAM locations are addressed by MWMGCFLD00-09. In local-addressing mode, the five RAM address MSBs (LOCADDR00-04) are supplied by the local storage address register (STORADDR00-04), and the five RAM address LSBs (LOCADDR05-09) are supplied by MWMGCFLD05-09. Therefore, in local-addressing mode, one of 32 local storage RAM "partitions" is addressed by the contents of the local storage address register (STORADDR00-04); and MWMGCFLD05-09 are an index, addressing one of 32 locations in the addressed partition.

When MWRAMODE is not asserted, MWMGCFLD 00-07 are selected as the eight local storage RAM address MSBs (LOCADDR00-07). The two address LSBs (LOCADDR08-09) do not go through the RAM mode multiplexer, but are always asserted by MWMGCFLD08-09. The RAM is now addressed in global-addressing mode.

When MWRAMODE is asserted, the contents of the local storage address register (STORADDR00-04) and MWMGCFLD05-07 are selected as the eight local storage RAM address MSBs (LOCADDR00-07). The RAM is now addressed in local-addressing mode.

5.3.15 Local Storage Address Register

The local storage address register (LSAR) is a 5-bit register that supplies the five local storage RAM address MSBs in local-addressing mode. The LSAR is initially loaded from MWMGCFLD00-04 at CLK4 time with MWSKIPFLD00, 02-04 = 20.

The primary function of the LSAR is to allow the local storage RAM to be address-organized into 32 partitions, each partition having 32 locations. This simplifies queue manipulation functions of the port. Changing the contents of the LSAR changes the five MSBs of the local storage RAM address, and accesses a different partition. The five local storage RAM address LSBs are always addressed by MWMGCFLD05-09 of the microword, allowing any location within a partition to be addressed without reloading the LSAR.

5.3.16 Skip Condition Field Decoder

The skip/condition field decoder (COND/SKIP) decodes the MWSKIPFLD00-04 field, CRAM bits 43-47, to determine which of the field functions is to be executed. All of the functions are internal to port microprocessor operation. The skip functions, encoded in MWSKIPFLD01-04, are given in Table 5-10, as are the condition functions, encoded in MWSKIPFLD00, 02-04.

Table 5-10 Skip/Condition Function

MWSKIPFLD 01-04	Function (CCMUX Input)
00	CCCBUSAVAIL
01	CCGRNTCSR
02	CCFEQ0
03	CCCSRCHNG
04	CCEBPARERR
05	CCRCVRBUFAFUL
06	CCRCVRBUFBFUL
07	CCXMTRATTN
10	CCEBUSRQST
11	CCINTRACTIVE
12	CCMBSIGN
13	CCMVRPARCHK
14	CCCBUSPARERR
15	CCPLIPARERR
16	CCCHANERR
17	CCCBLSTWD

Table 5-10 Skip/Condition Function (Cont)

MWSKIPFLD 01-04	Function (CCMUX Input)
20	LOADSADREG
21	SELMBUSFLD
22	RDLOCALMEM
23	LDLOCALMEM
24	SELCNSTFLD

5.3.17 Microprocessor Control Logic

The microprocessor control logic controls all of the timing functions of the port microprocessor. It contains the necessary control logic to:

1. Write the RAM address register (RAR).
2. Write and read/verify the control RAM (CRAM).
3. Read the latch address register (LAR).
4. Generate RUNCLK1, RUNCLK2, RUNCLK3, and RUNCLK4 from CLK1, CLK2, CLK3, and CLK4 respectively.
5. Start and stop the port microprocessor in an orderly way.

5.4 MICROCODE

This section contains the flow diagrams with corresponding descriptions of the NIA20 microcode, which performs the following major functions: initialization idle loop, receive, and transmit and local command. (See Figures 5-20 through 5-23.)

5.4.1 Initialization

Figure 5-20 is a flow diagram of the initialization microcode routine. When the port is powered-up, there is no valid microcode in the CRAM. Valid microcode must be loaded and started when the port is in the uninitialized state. At the start of the initialization sequence, the port microcode performs self-tests. These tests check the Am2901 addressing and operation, the Am2910, and the local store (LS) addressing. If any of these tests fails, the microcode loops on the FAIL.SELF-TEST location in the fatal error table.

The microcode next sets the LS -- clearing specified locations and loading others with bit masks and field masks. When the NIA20 is being initialized, the port driver sets the KL interface data channel to transfer three words of the port control block (PCB) over the CBus and into the port. A channel command word (CCW) is set to transfer PCB words containing the PCB base address, the physical interrupt assignment (PIA) - physical interconnect (PI) level assignment, and a reserved word from KL memory to the data channel.

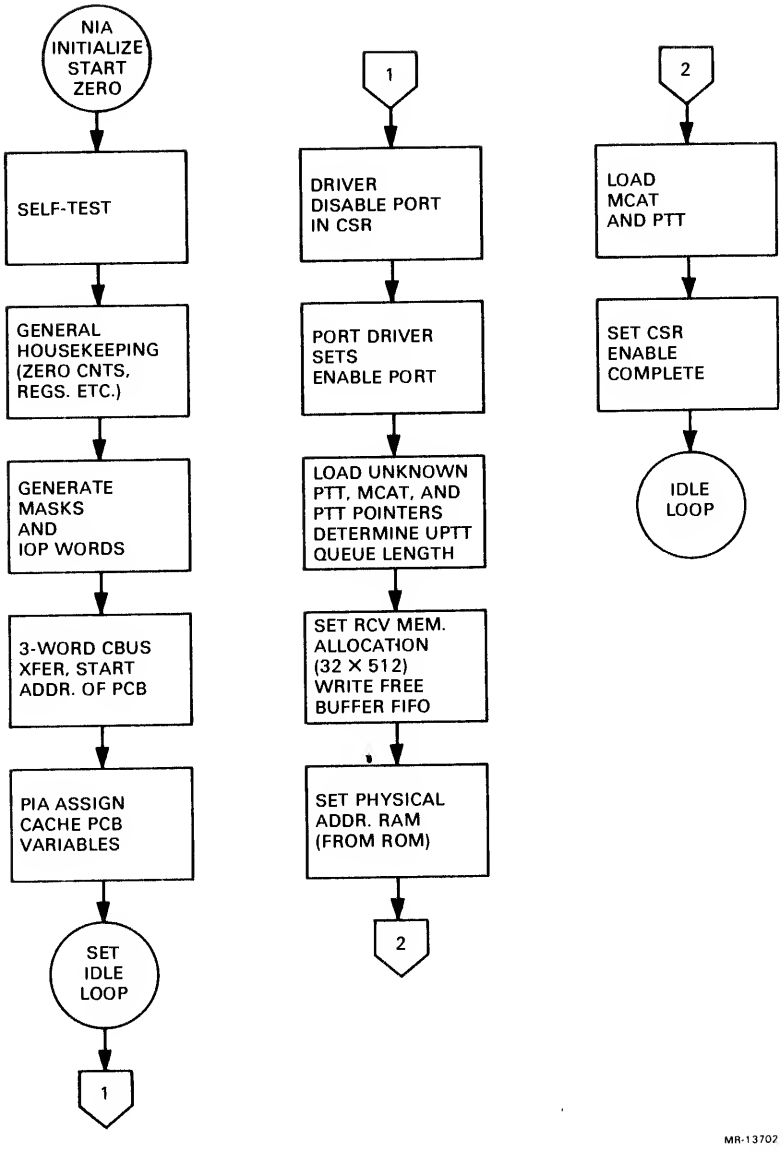


Figure 5-20 Initialization Microcode Flow Diagram

In the next sequence of this routine, the port starts the channel with a CBus START microprocessor command and reads the contents of these words into the LS. The port now has its PIA and PCB base address and can request interrupts. After stopping the CBus, the port writes the physical addresses of the queue interlocks, FLINKS and BLINKS, error words, and the CCW into specified LS locations. These addresses are determined by the port by incrementing the PCB base address value it received in the CBus transfer.

The microcode then writes the basic IOP function control words for standard (40 + 2n), examine, deposit, and examine and increment interrupts into the LS. Vectored interrupts are not used.

Finally, the local store address register (LSAR) offsets to the port cache addresses that are allocated in the LS. These caches hold packet or command information and the microprocessor's state while the packet or command is being processed. This information includes the queue entries FLINK and BLINK, the interlock word PCB addresses, the buffer header address, and the current buffer segment descriptor (BSD). The caches are listed in Table 5-11.

Table 5-11 Cache Base Addresses

Cache	Base Address	
RCVR	700	LSAR base offset to enter RCVR cache
XMIT	640	LSAR base offset to enter XMIT cache

The cache locations are referenced by the LSAR offsets listed in Table 5-12. These offsets are contained in the microword magic number field and can be referenced by more than one offset. For example, offsets CQC.FLAGS, CQC.STATE, and CQC.PAK all have a value of 4. Therefore, a mask is used to isolate a particular bit or field in the cache word. The bits, fields, and masks for CQC.FLAGS, CQC.STATE, and CQC.PAK are defined in an LS block called the command queue status block flag word. Refer to Table 5-13 for a list of these words and the LS mask address. For example, CSTATE.MSK (LS address 173) is a mask for cache state bits 20-23 in the CQC.STATE field.

Table 5-12 Local Store Address Register Command Status Block Offsets

Name	Offset Value	Definition
CQC.LSAR_OFF	0	LASAR offset of this block
CQC.INTRLK	1	Address of PCB queue interlock word
CQC.FLINK	2	Address of PCB queue FLINK word
CQC.BLINK	3	Address of PCB queue BLINK word
CQC.FLAGS	4	Flags word
CQC.STATE	4	Cache state
CQC.PAK	4	Packing mode
CQC.OPCODE	5	Operation code word
CQC.QUEUE	6	FLINK address of this command
CQC.RSVD	7	Command reserved word
CQC.USEDBUF.0	7	Number of 512.buf used to store received frame
CQC.HXCTID	10	Transaction ID word 1
CQC.LXCTID	11	Transaction ID word 2
CQC.XCTLEN	12	Transaction length
CQC.SNDNAM	13	Send buffer name
CQC.MAINTID	13	ID, REQID maintenance ID word
CQC.BLDRSP.STS	14	Status field for build response
CQC.CODREV	14	ID, REQID microcode revision level word
CQC.SEND1	15	Sender's station address bytes 0-3
CQC.HDEST	15	Destination station address bytes 0-3
CQC.SEND2	16	Sender's station address bytes 4 & 5
CQC.LDEST	16	Destination station address bytes 4 & 5
CQC.BHDBASE	17	Buffer header descriptor address
CQC.BHDLEN	20	

Table 5-12 Local Store Address Register Command Status Block Offsets (Cont)

Name	Offset Value	Definition
CQC.SEGBAS	21	BSD segment base address
CQC.BSDBASE	22	Buffer segment descriptor address
CQC.BSDLEN	23	
CQC.FBSD	24	
CQC.NBSD	25	Next BSD address
CQC.BSDRES	26	
CQC.BHDRES	27	
CQC.CASAVE	30	
CQC.BCOFF	31	
CQC.BLDRSP.OPC	31	Saved operation code word from response queue entry
CQC.BCRES	32	
CQC.WCRES	33	
CQC.PAKLEN	34	Transfer length in bytes
CQC.PAKRES	35	Number of words left to transfer
CQC.FRQUE	36	Datagram free queue for no-build response
CQC.PR_TYPE	37	Protocol type from received frame

Table 5-13 Command Queue Status Block Flag Word

Name	LS RAM Address	Bit	Definition (when bit is on)
CC.STATE	206	20-23	Cache state word 0=FREE 1=MPKT 2=SPKT 3=XMIT 4=NEW
CS.FORMAT	210	18-19	Packet format 0=Industry compatible 1=Reserved 2=Reserved
CS.DIRECTION	123	17	Transfer direction 0=CBUS <=PLI 1=PLI <=CBUS

At this point in the microcode routine, the idle loop is entered. The port is still in the unintialized state. The packet or command state is maintained primarily in flag and status bits. The cache can be in one of the following states:

1. FREE - Cache not currently allocated
2. MPKT - Mover has packet (unused)
3. SPKT - Suspend packet (await processing completion)
4. XMIT - Ready to transmit
5. NEW - Cache just built.

5.4.2 Idle Loop

Figure 5-21 is a flow diagram of the idle loop microcode routine. This routine is performed after initialization has been successfully completed and all port microcode functions start from the idle loop. When a function is completed, the microcode returns to the idle loop. If the port conducts a power-up or reset, the idle loop is entered from an uninitialized state.

A standard interrupt request (40 + 2n, PI level 01-07) is the initial event performed in establishing the idle loop. The microprocessor compares the INTREQ mask from the LS with the INTREQ flag. If the flag is set, the FEQO (ALU function=0) condition code is asserted through the condition code multiplexer with a condition jump to GEN.INTERRUPT.

If the contents of the CSR have changed, the next idle loop function examines the CSR by executing a conditional jump to subroutine CSR.SRV. The jump will be executed when the CSR changed condition code is asserted through the condition code multiplexer.

The microprocessor saves its copy of the CSR (CSRCPY) in a temporary register Am2901 RAM location T1 and reads the new CSR contents into T0 and CSRCPY. It then compares the old and new CSRCPY for a change in CSR bits 30 (DISABLE) or 31 (ENABLE) or the setting of CSR27 (CMD QUEUE AVAIL). If either bits 30 or 31 have not changed or bit 27 was not set, the microprocessor returns to processing of the idle loop routine.

If DISABLE has been set and the port is uninitialized, the microprocessor sets CSR12 (disable complete), and FLAGS bit 22 (disable flag). If DISABLE is set when the port is in the enable state, the microprocessor performs the following:

Sets CSR12 (disable complete)
Clears CSR13 (enable complete)
Clears CSR31 (enable)
Clears FLAGS bit 21 (enable flag)
Sets FLAGS bit 22 (disable flag)
Disables the NIA module.

The port has now completed the transition from the enabled to the disabled state.

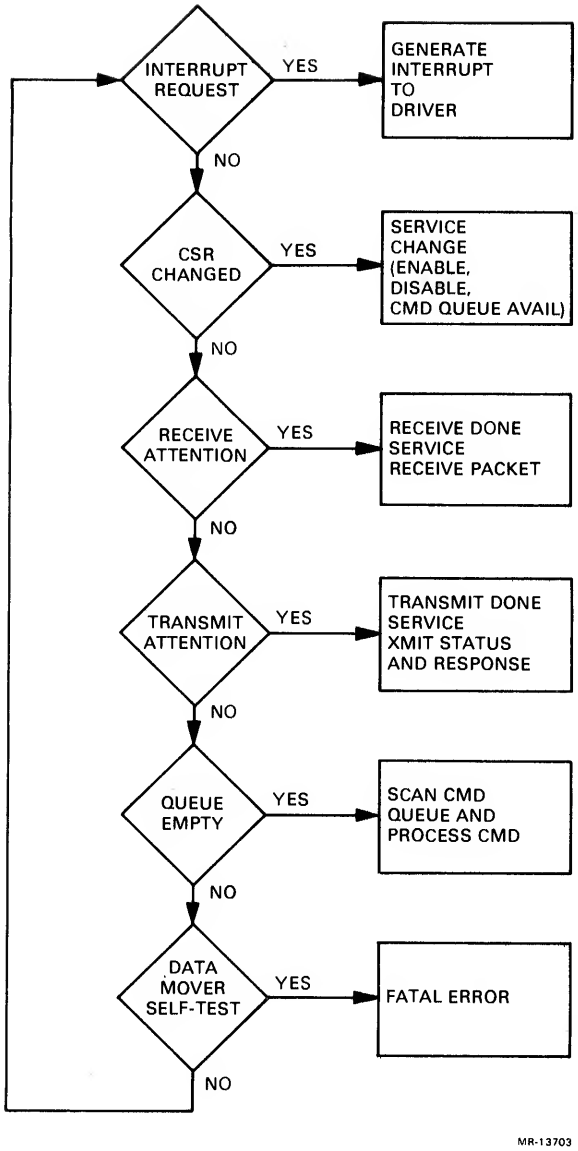


Figure 5-21 Idle Loop Microcode Flow Diagram

If DISABLE has been cleared, the microprocessor clears CSR12 (disable complete), the FLAGS bit 22 (disable) if CSR31 (enable) is not set, and the disable bit in the system state word. The microprocessor returns to continue idle loop processing.

If ENABLE is set and the port was not in disable, the microprocessor returns to continue idle loop processing.

If ENABLE is set and the port was disabled, the microprocessor performs the following:

- Set FLAGS bit 23 (run flag)
- Set CSR13 (enable complete)
- Clear CSR12 (disable complete)
- Clear CSR30 (disable)
- Clear FLAGS bit 22 (disable flag)
- Set FLAGS bit 21 (enable)
- Clear system state word bit 22 (disable)
- Return to continue idle loop processing
- Cache PTT
- Cache MCAT
- Write ROM address to RAM
- Load the free buffer list (FIFO) with addresses used during NIA packet reception
- Set the receive buffer to 32 x 512 (other available byte sizes are 64 x 256 and 16 x 1024)
- Enable the NIA link to receive packets by setting ENABLE LINK in the link control register.

The port is now in the enabled state. If ENABLE is cleared, the microprocessor clears CSR13 (enable complete), clears FLAGS bit 21 (enable), disables the NIA, and then returns to idle loop processing.

If the CMD QUEUE AVAIL bit is set, it is cleared by the microprocessor. When the ENABLE flag is not set, the microprocessor returns to continue processing the idle loop. If the ENABLE flag is set, the microprocessor sets the queue available in the LS location port queue status (PQS).

The next idle loop function looks for a RECEIVE ATTENTION. If the receive attention condition code is asserted, the microprocessor executes a conditional jump to RECEIVE.DONE.

When the microprocessor detects a TRANSMIT ATTENTION, the transmit status of any outgoing packet is checked. If a transmit attention is present, a conditional jump to TRANSMIT.DONE is executed. An XMIT ATTN signal indicates a completed or an aborted transmission, which enables the transmit attention condition code through the condition code multiplexer.

The last idle loop event in the routine causes the microprocessor to check the queues for a command queue entry by comparing the LS location PQS contents with the ALL.QUES mask. If all command queues are not empty, a jump to SCAN.QUEUES is executed. If all the queues are empty, the FLAGS bit 8 (CACHE.FULL) is tested and, if set, a jump to SCAN.QUEUES is executed.

If a jump to SCAN.QUEUES is executed, the next command to process is selected. If a cache is ready for processing, the microprocessor activates the cache and jumps to SEND.PACKET. If none of the caches is ready for processing, the code looks for a command queue entry to process. If a command queue entry is available, the microprocessor jumps to BUILD.CACHE to start processing the new command.

5.4.3 Receive

Figure 5-22 is a flow chart of the NIA receive microcode routine. This routine is called when the RCV ATTENTION signal is asserted from the idle loop. The RCV ATTN signal indicates that an incoming frame has passed the hardware address filtering process and storage space is available in the NIA receive buffer.

The microprocessor then reads the receive status, and, if any receive errors are detected, data transfer is stopped, an error status mask is built in the response status field, and the error counter is incremented.

When a free buffer parity error is detected, the NIA module is disabled and a jump to PLCRPE (location 7766) halts the operation. If no free buffer error is detected, the following are performed:

- Set the error mask
- Increment the receive fail counter
- Reset RCV ATTN
- Return to the idle loop.

If there is no free buffer parity error or no free buffer error or no other error present, the number of used buffer entries is read from receive status for the valid number of buffers. If a valid number of buffers cannot be obtained, the corresponding error event count is incremented and a return to the idle loop executed.

The receive cache is latched after obtaining the valid number of buffers. After successfully obtaining the receive cache, the used buffer list is read the correct number of times, as indicated in the receive status. With the selection of the first receive buffer pointer in the used buffer list, the destination address is read and then filtered. The following bytes are read from the NIA receive buffer:

1. Destination
2. Source
3. Protocol type
4. Data
5. CRC.

The filtering process determines if the destination address is a physical, multicast, or broadcast address that matches any of the listed addresses in the multicast address table (MCAT). If no match of the destination address in the received frame results from this filtering process, the no-match discard frame counter is incremented and the frame discarded. If the filtering process matches an address, the source address and protocol type (PT) are cached. If the protocol type is not enabled, an unknown protocol type is used before performing a delink queue operation.

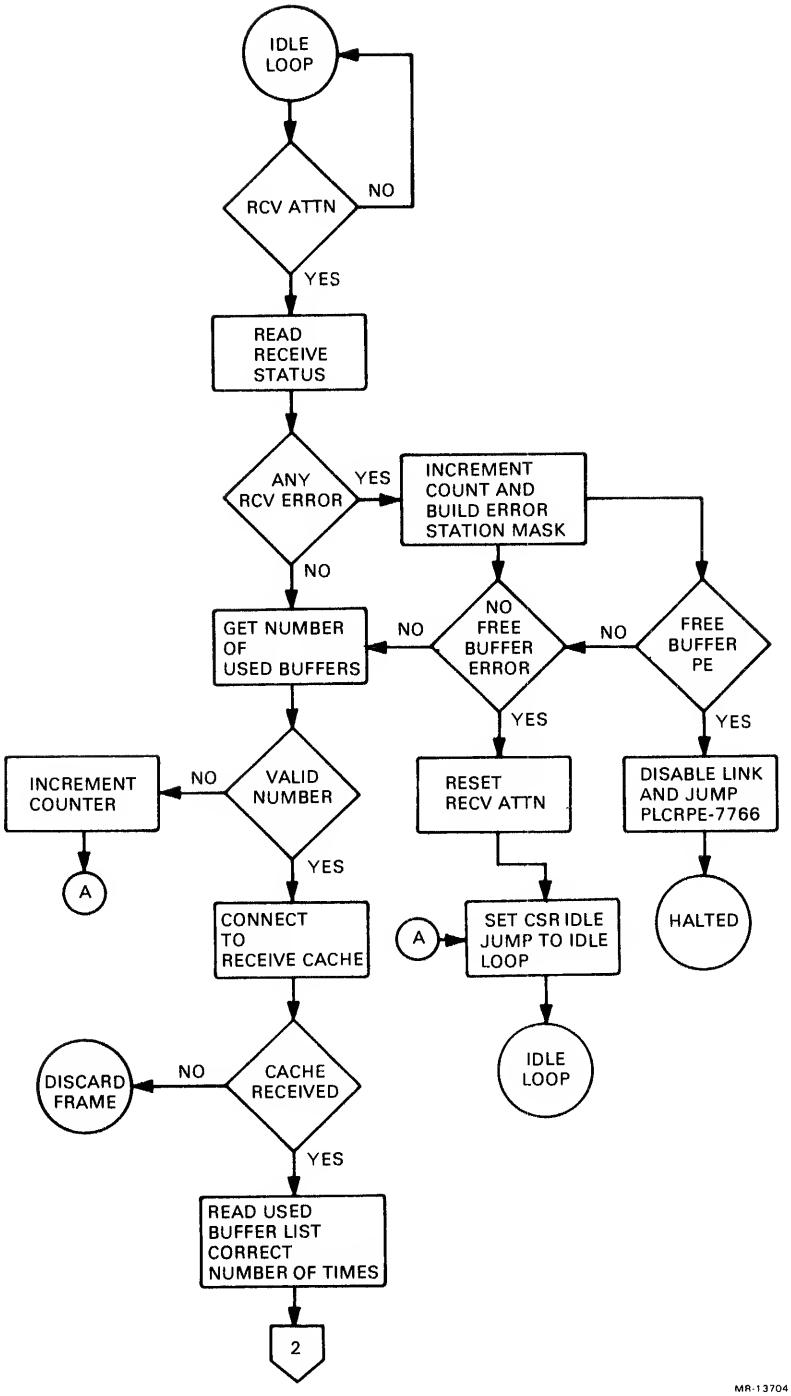


Figure 5-22 Receive Microcode Flow Diagram
(Sheet 1 of 6 sheets)

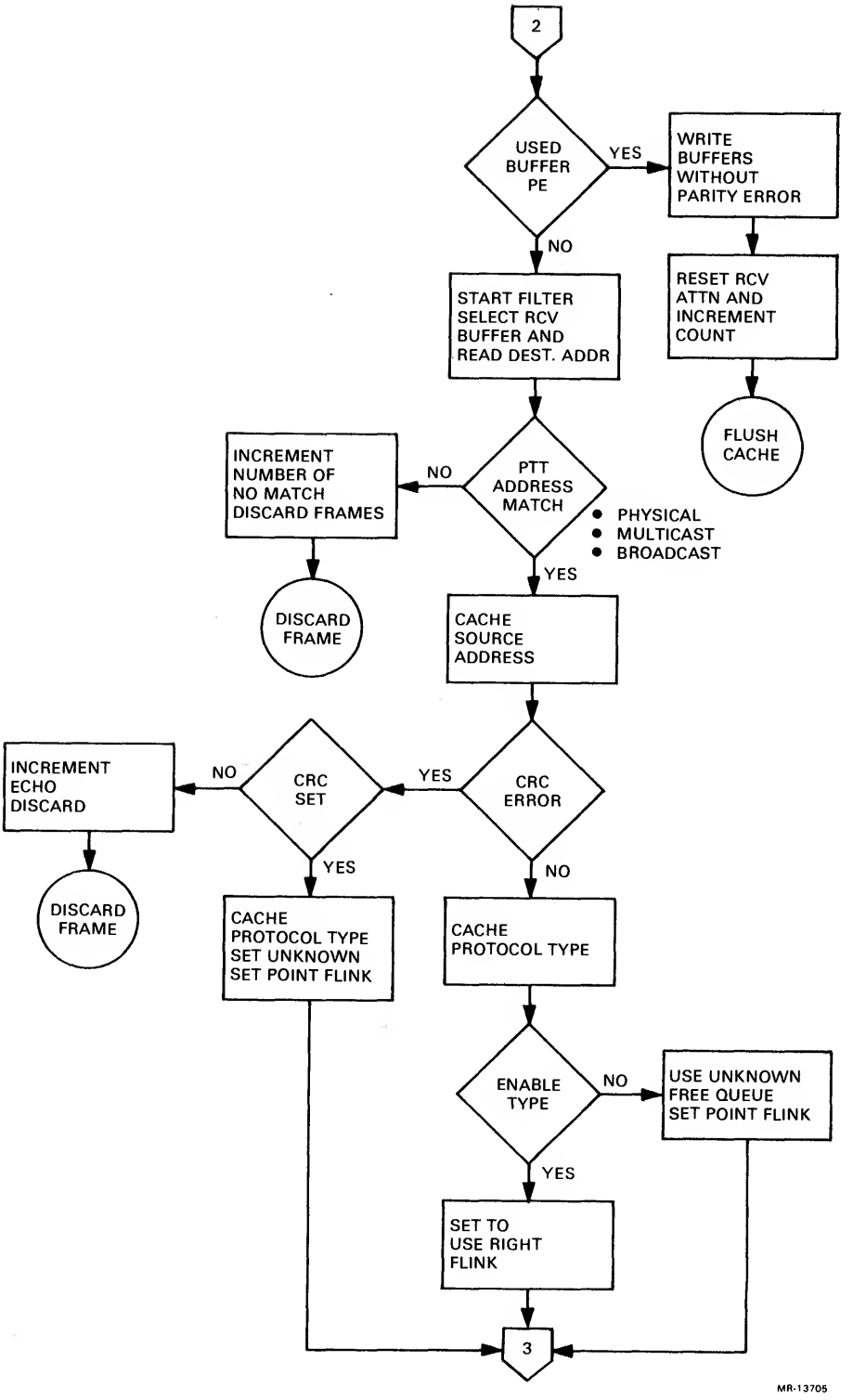


Figure 5-22 Receive Microcode Flow Diagram (Sheet 2 of 6 sheets)

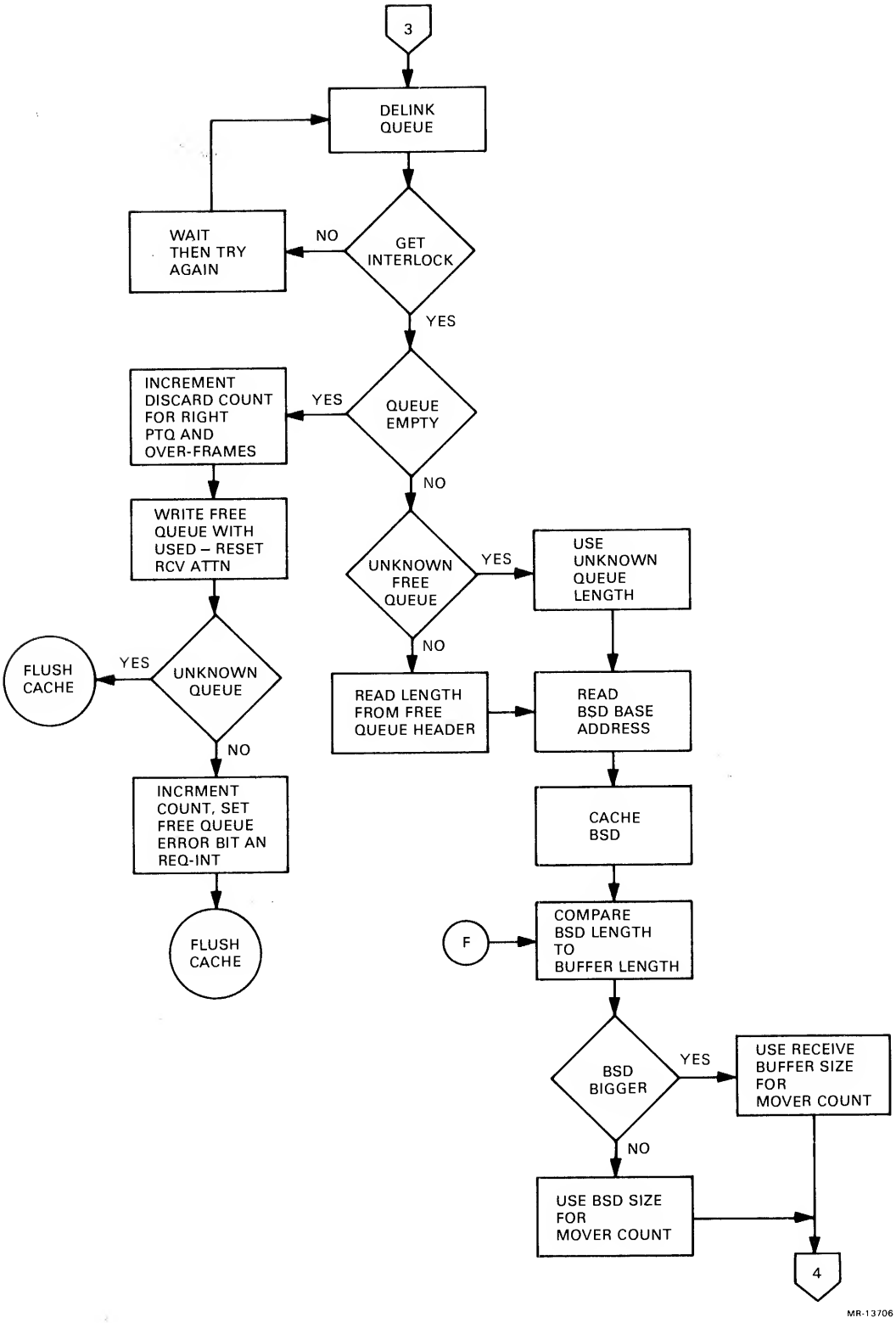


Figure 5-22 Receive Microcode Flow Diagram
(Sheet 3 of 6 sheets)

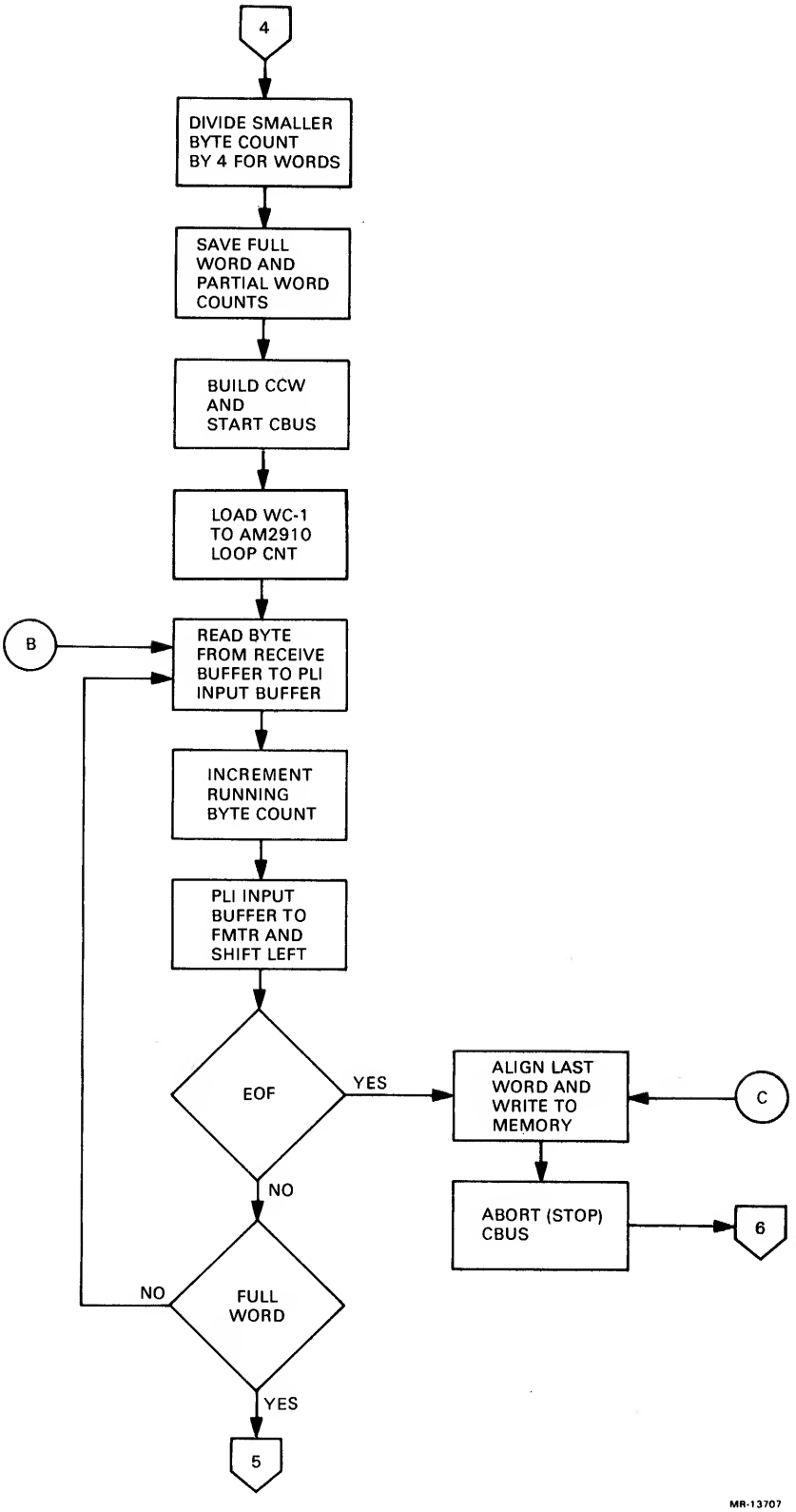


Figure 5-22 Receive Microcode Flow Diagram (Sheet 4 of 6 sheets)

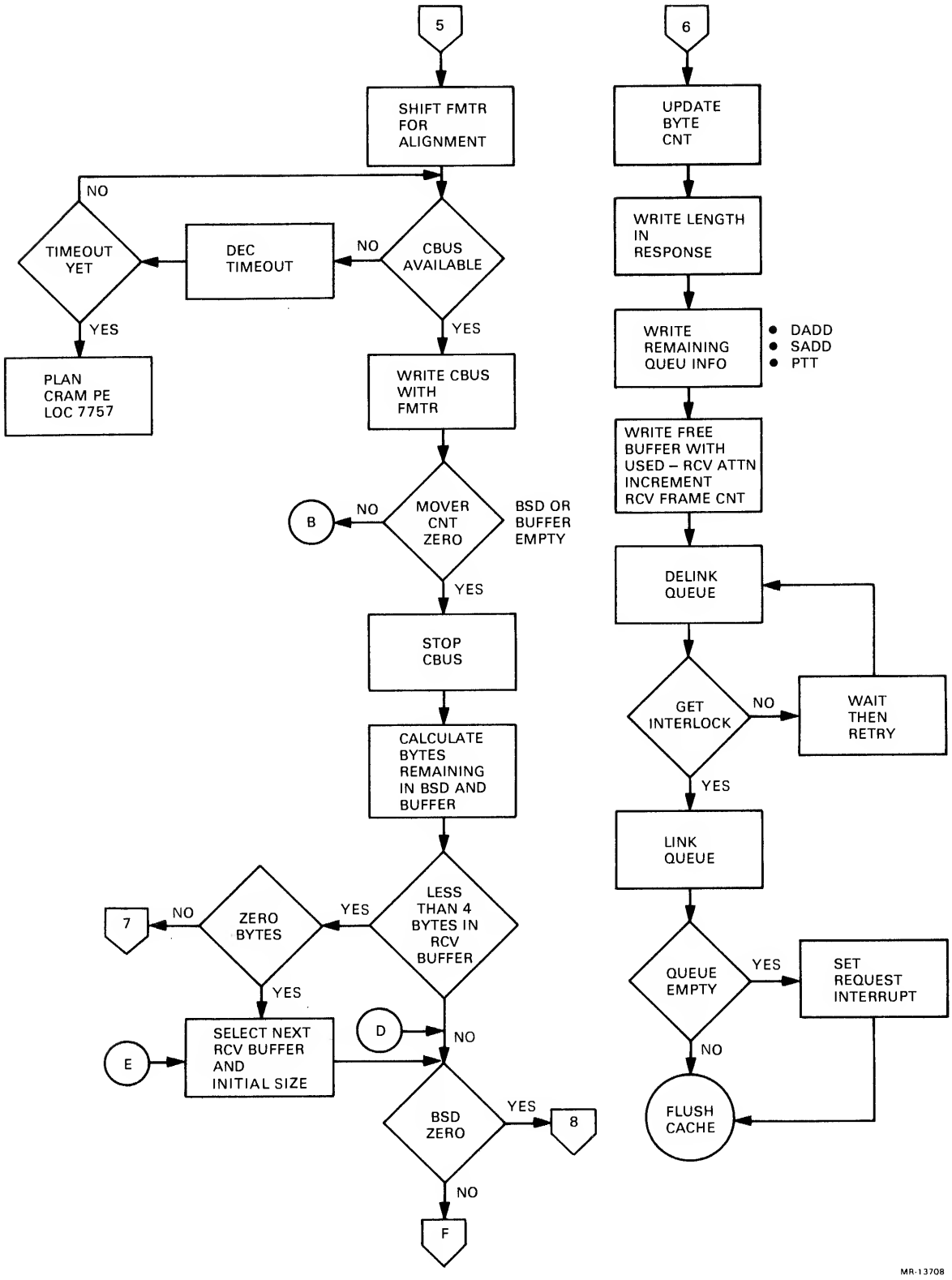


Figure 5-22 Receive Microcode Flow Diagram
(Sheet 5 of 6 sheets)

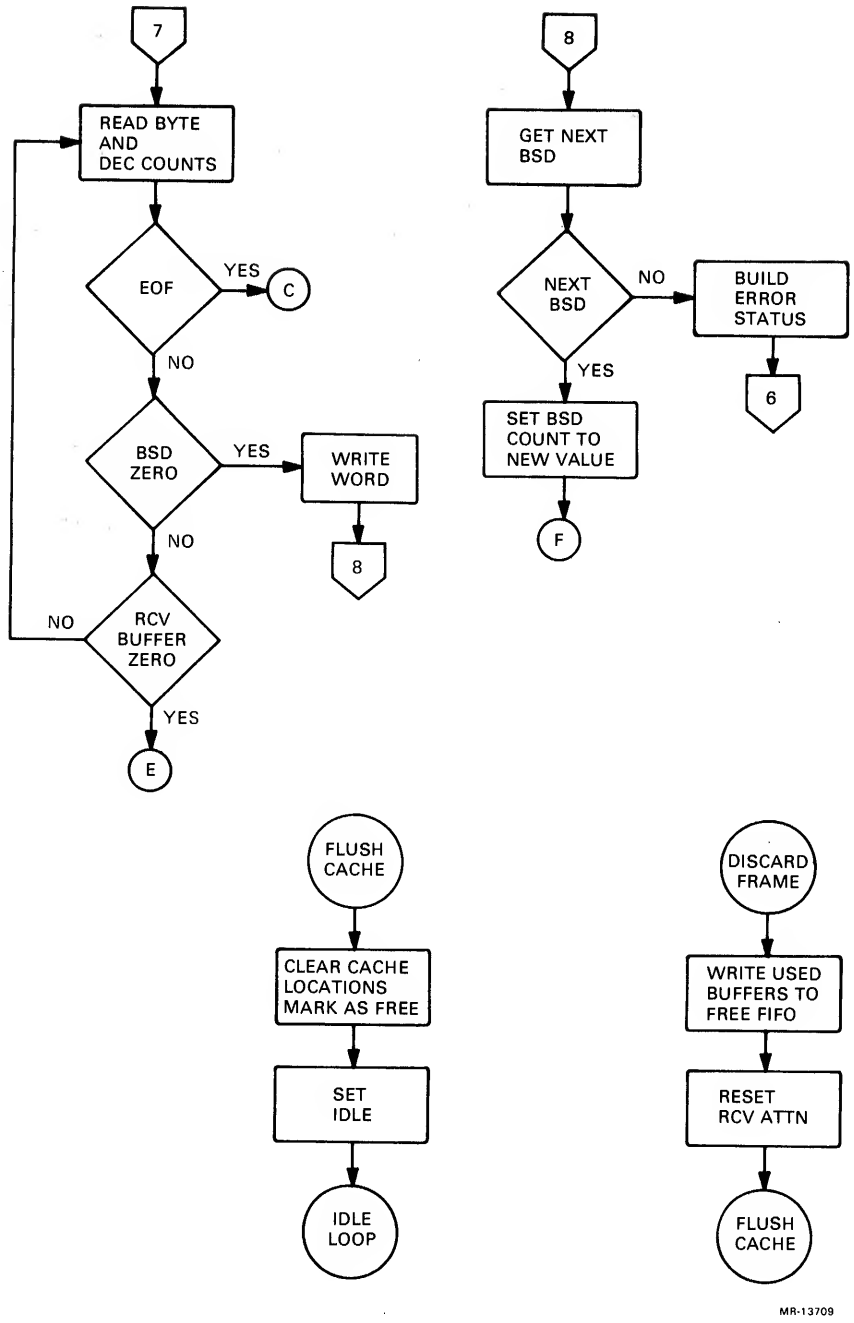


Figure 5-22 Receive Microcode Flow Diagram (Sheet 6 of 6 sheets)

When connecting to the cache fails, a discard frame subroutine is entered and the frame discarded. In the discard frame subroutine, the used buffers are written into the free buffer list FIFO, the RCV ATTN reset, and a flush cache subroutine performed. A flush cache subroutine clears the cache locations and marks the cache as free before returning to the idle loop.

A delink queue operation removes an entry from a free queue by manipulating entry FLINKS and BLINKS in KL10 memory. The port first generates an EBus interrupt to access an interlock word. If the queue was interlocked, the port waits for 512 cycles (163.84 μs) and then jumps back to the start of the allocate queue subroutine.

If the queue was empty, the following steps occur in sequence:

- Increment respective protocol type discard count
- Write used buffers to free list
- Reset RCV ATTN
- Set the CSR free queue error bit, if this is a known PT
- Set interrupt request flag
- Execute a flush cache subroutine (whether queue empty or known PT)

When the free queue is a known PT, its length is read from the queue header. If the queue is an unknown free queue, its length is read from the port control block (PCB).

The BSD is then cached and its length compared to the receive buffer length to determine which of the two lengths is the smaller. A functional sequence then follows, including:

- Use the smaller of the BSD/receive buffer lengths to set the data mover count
- Divide word count by four for a byte count
- Save full word/partial word counts
- Build channel command word (CCW)
- Start CBus
- Load word count minus one into the Am2910 microsequencer loop counter
- Read receive buffer to the port logic interface (PLI) input buffer incrementing the running byte count
- Shift each 4 bytes (full word) into the formatter
- Write each full word to host using CBus until the BSD or the buffer is empty (Loop count = zero)

When the loop count is zero, the loop stops and the total bytes read is subtracted from the BSD length and the receive buffer length. Any partial words that remain in the BSD or receive buffer are moved and the next BSD accessed, if the BSD equals zero. If the receive buffer equals zero, the next buffer is selected. A return to the beginning of the loop, where the smaller size buffer is selected, restarts the loop to repeat this transfer process.

Each time a 4-byte word is read from the receive buffer in the loop, a test for an end-of-frame (EOF) signal is conducted. The detection of an EOF stops the loop and aligns the format for the last word as it is written to the host over the CBus. With the parallel transfer completed, the CBus is then halted by the port and the following cleanup sequence occurs:

- Update byte counts
- Write the length, destination address, source address, and PTT to the response queue
- Write the used buffers to the free list
- Reset RCV ATTN
- Increment the receive frame count
- Link entry on response queue

When the entry has been successfully linked on the queue, the response queue available bit is set in the CSR, if the queue was empty and the interrupt request flag set. The cache is then flushed.

5.4.4 Transmit and Local Command

Figure 5-23 is a flow diagram of the transmit and local command microcode routine. This routine is started from the idle loop when a CSR change indicates a command queue available.

With the command queue available bit enabled in the CSR, the microcode scans the queue for command entries. If a command entry is present, the code disables the command queue available bit in the CSR, marks the queue available in the port queue status (PQS), and then returns to the idle loop. If no command queue available entry can be accessed, the subroutine returns to the idle loop.

Next, from the idle loop, if there is a queue marked, the microcode will try to delink an entry (all delinking of queue entries uses the same subroutine).

If the queue was empty, the microcode clears the PQS command available bit and returns to the idle loop. When the queue is not empty; the FLINK is saved in cache, the four-word queue header is read into cache, and the flag CC.XMIT is set to indicate CBus to PLI data transfers.

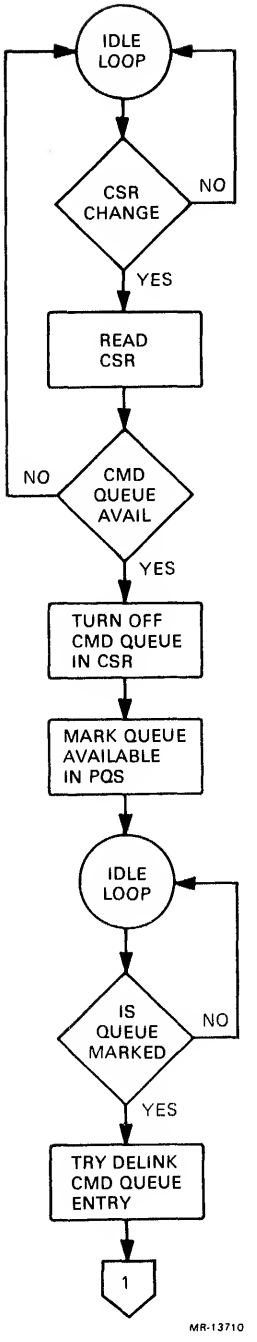


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 1 of 19 sheets)

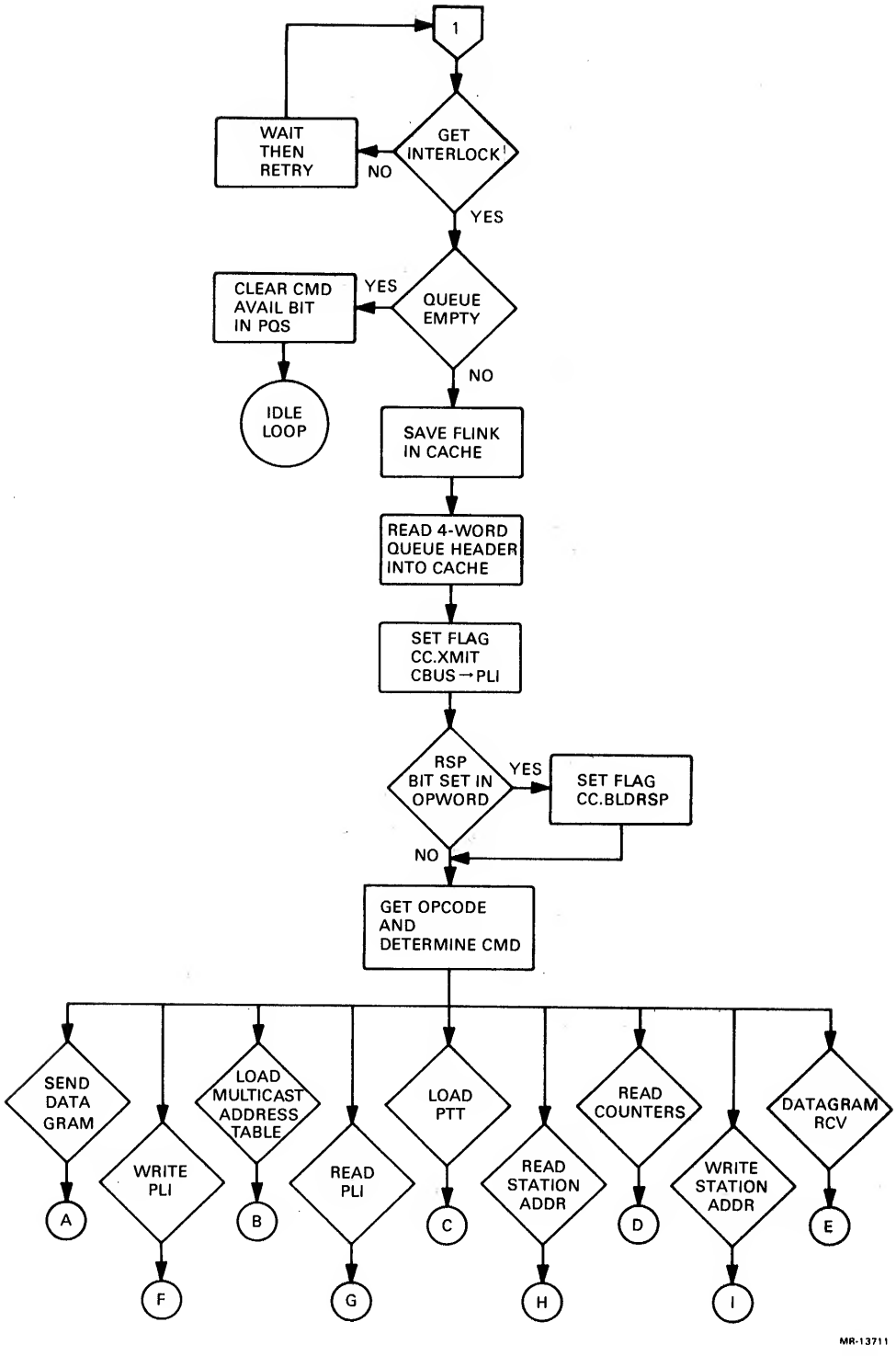


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 2 of 19 sheets)

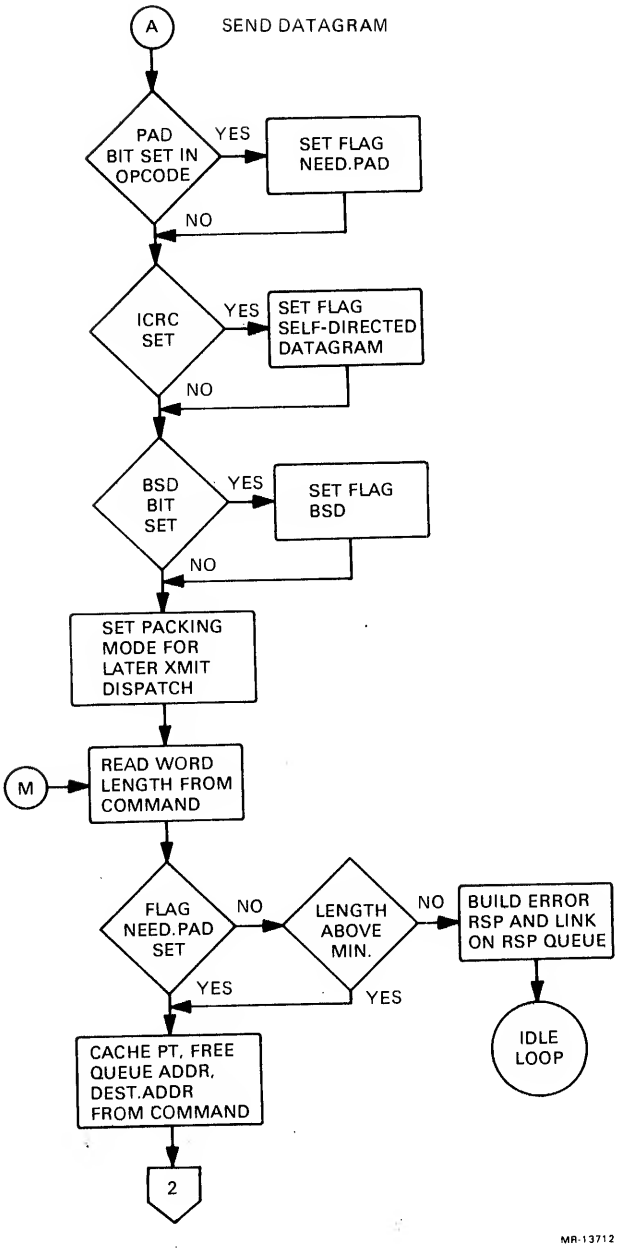


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 3 of 19 sheets)

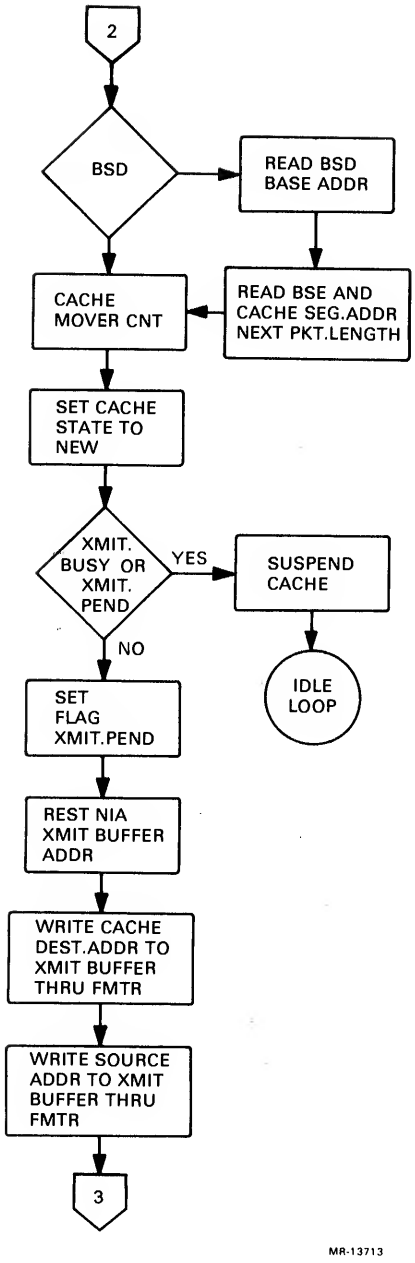


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 4 of 19 sheets)

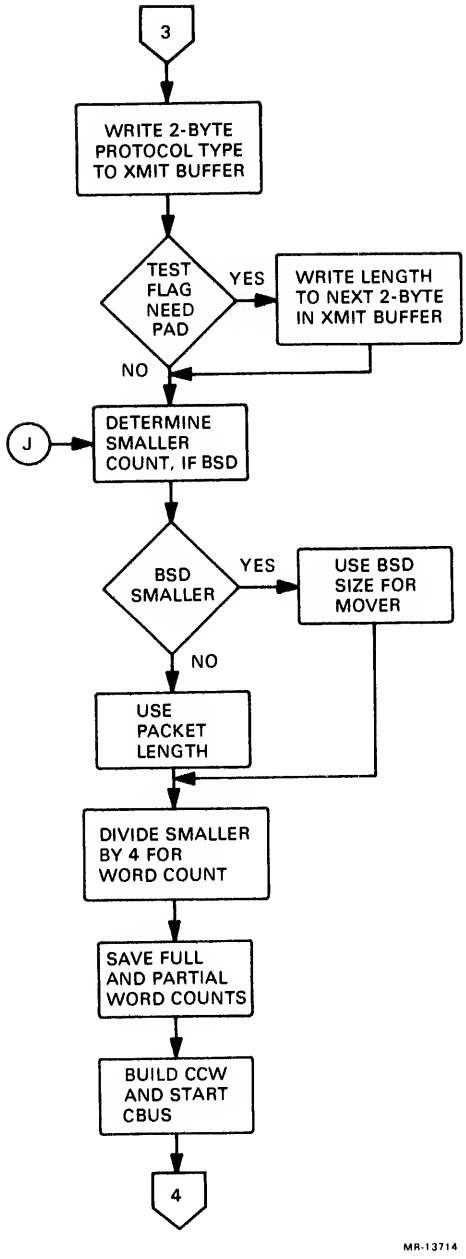


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 5 of 19 sheets)

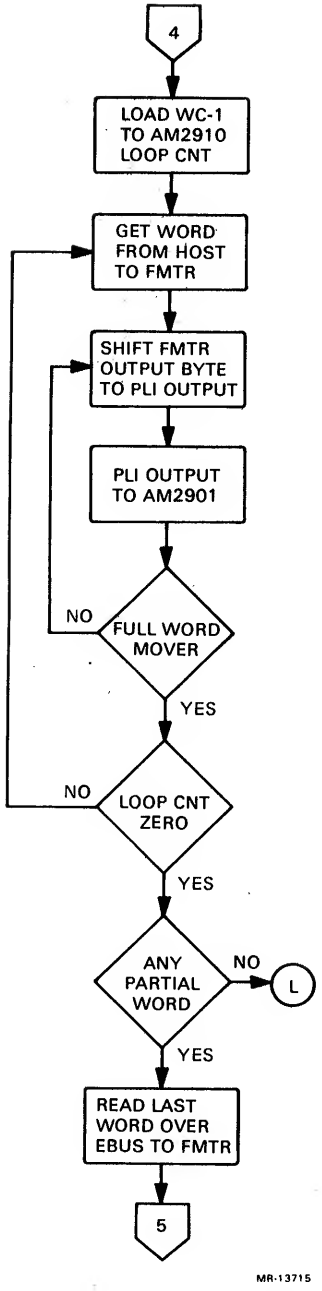


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 6 of 19 sheets)

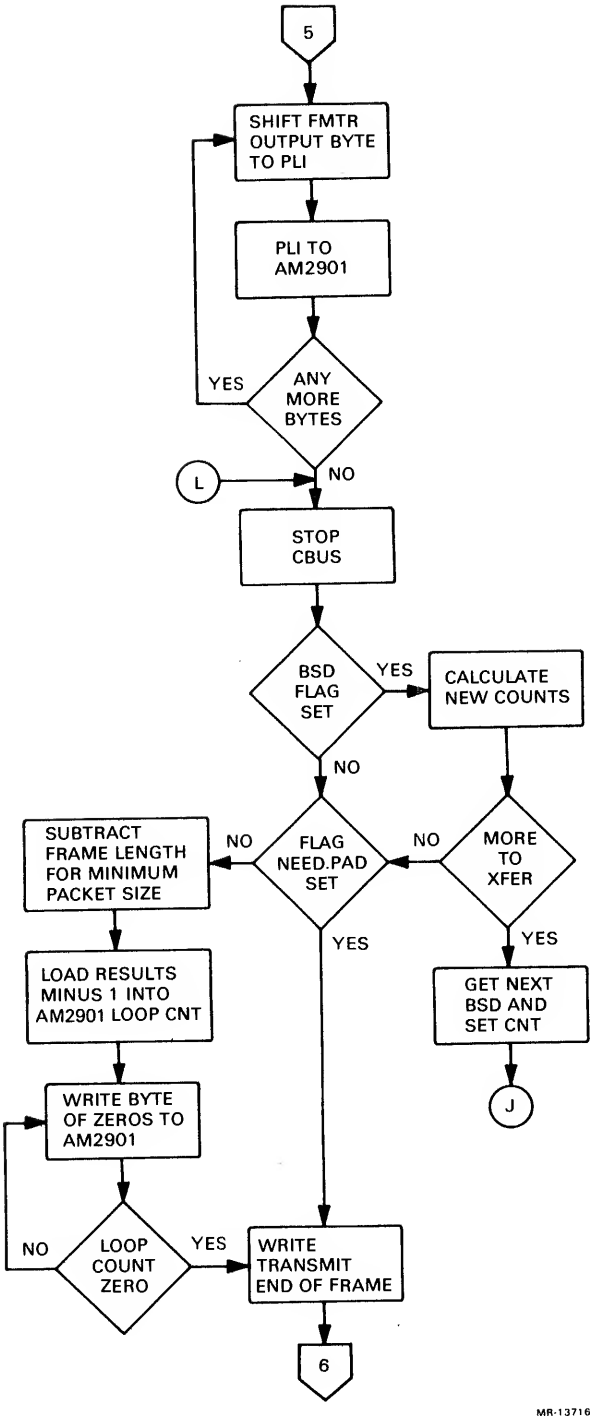


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 7 of 19 sheets)

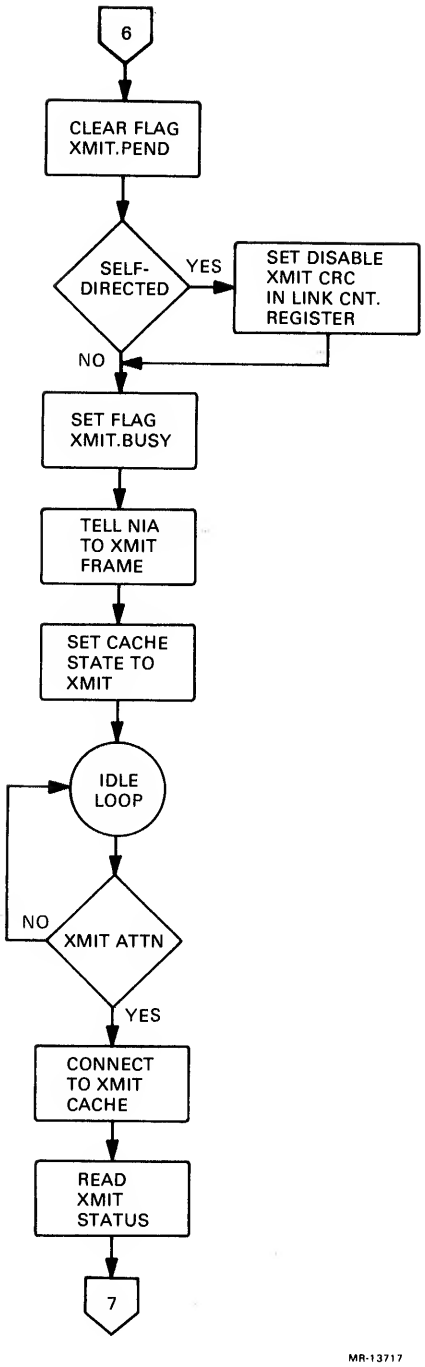


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 8 of 19 sheets)

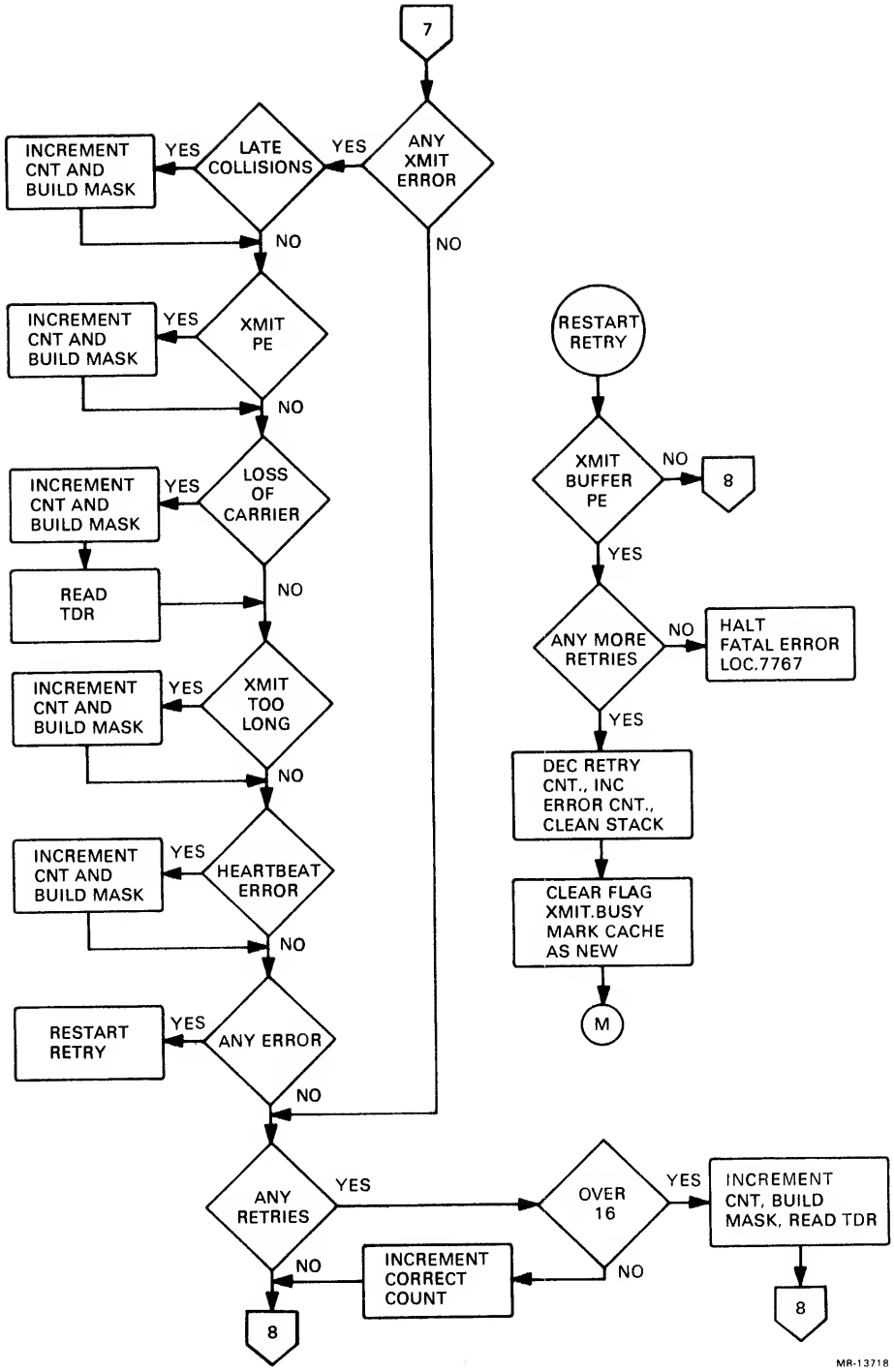


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 9 of 19 sheets)

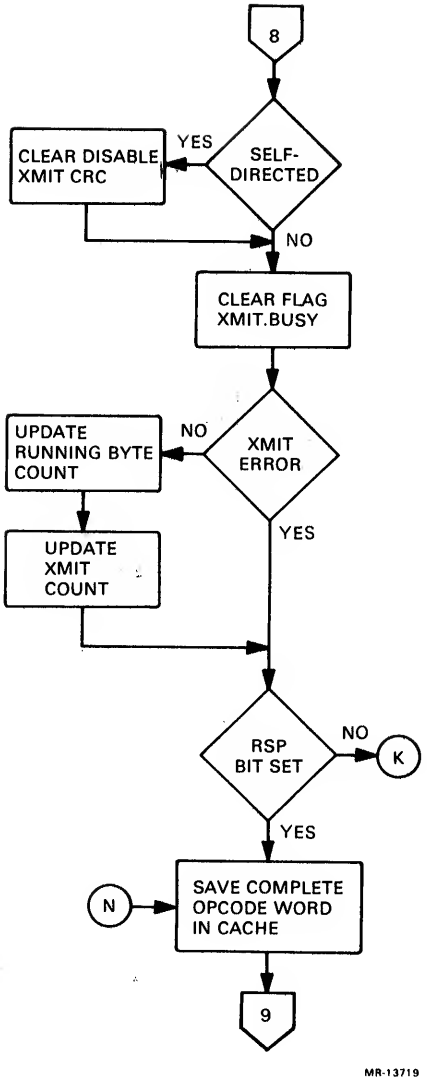


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 10 of 19 sheets)

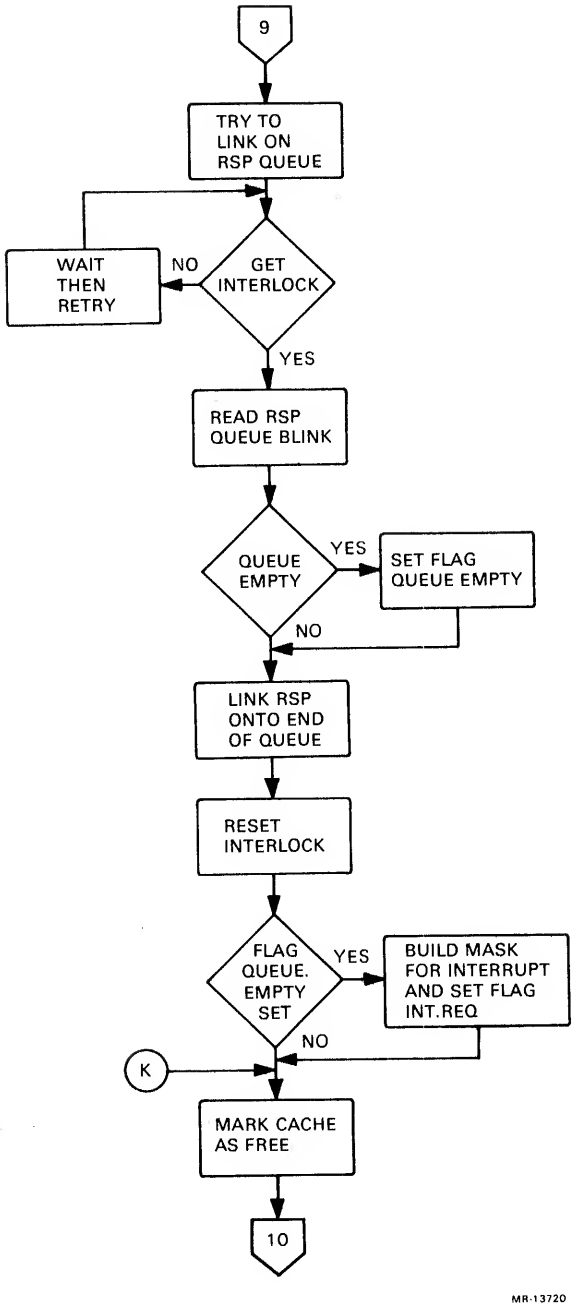


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 11 of 19 sheets)

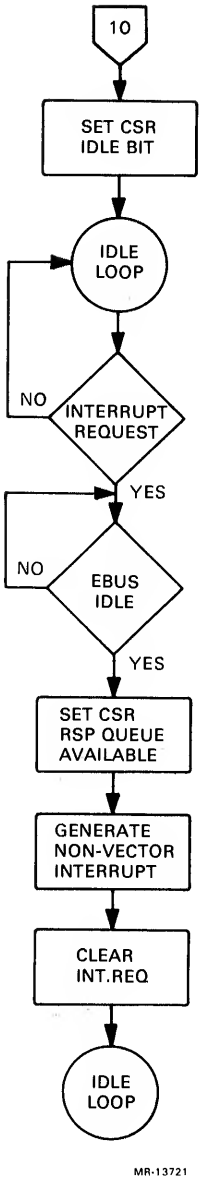


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 12 of 19 sheets)

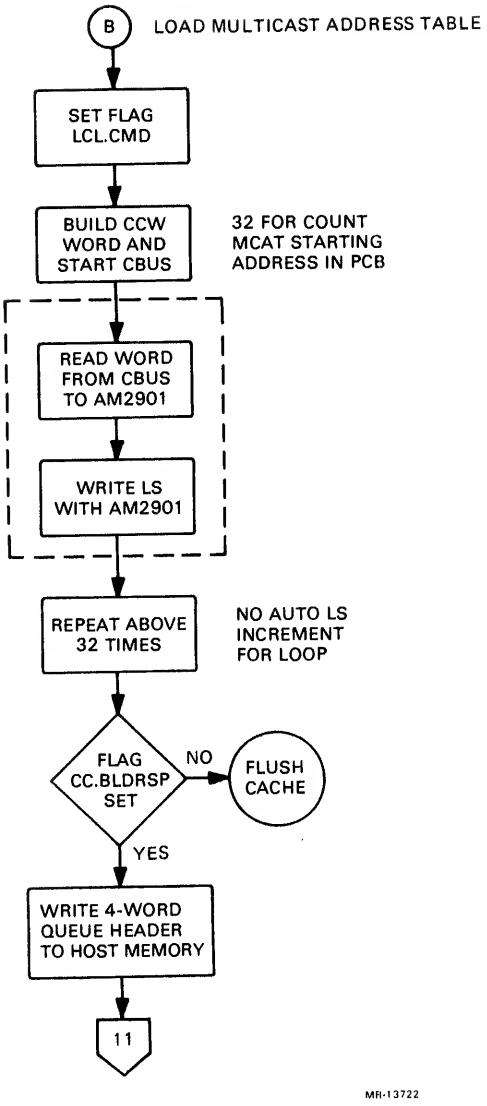


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 13 of 19 sheets)

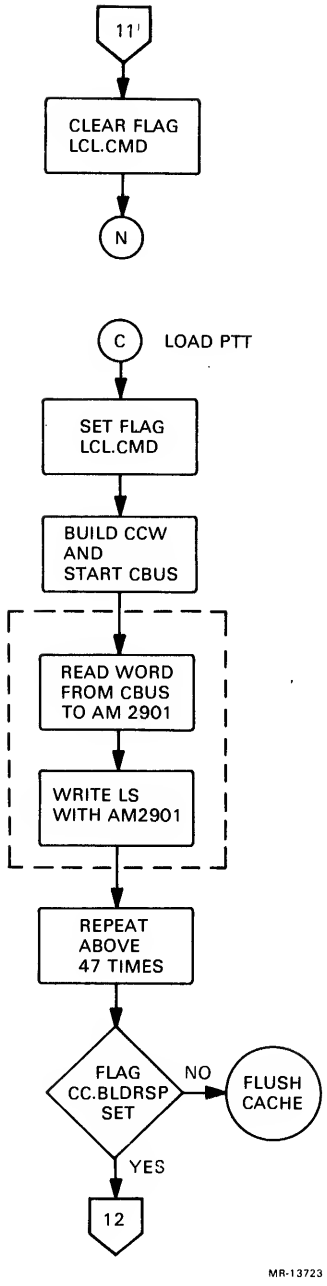


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 14 of 19 sheets)

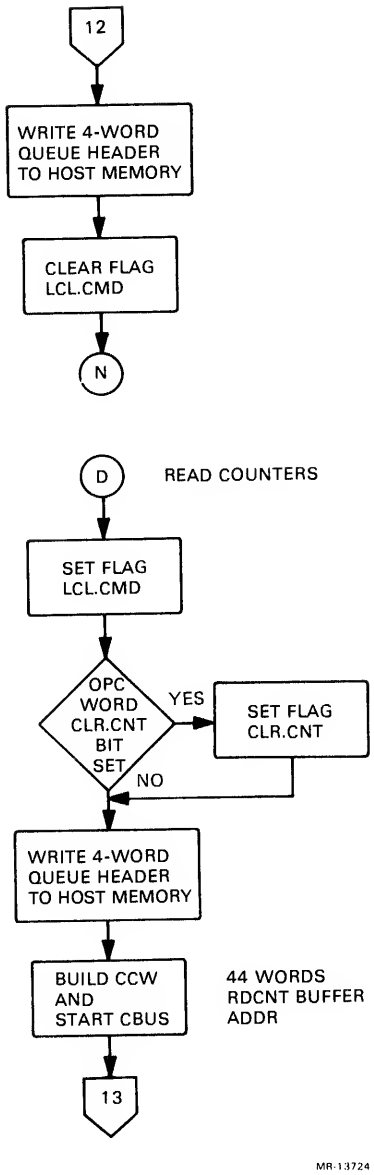


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 15 of 19 sheets)

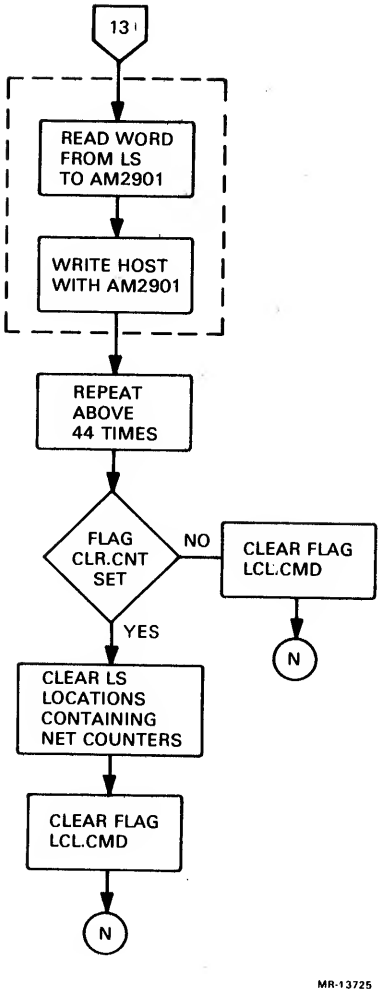


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 16 of 19 sheets)

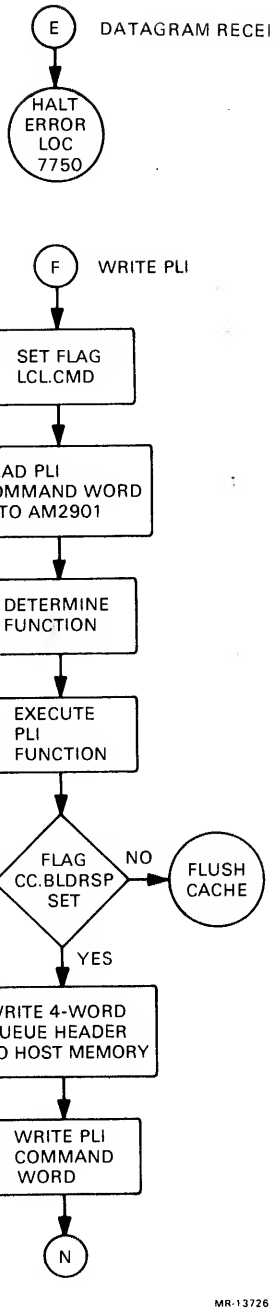


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 17 of 19 sheets)

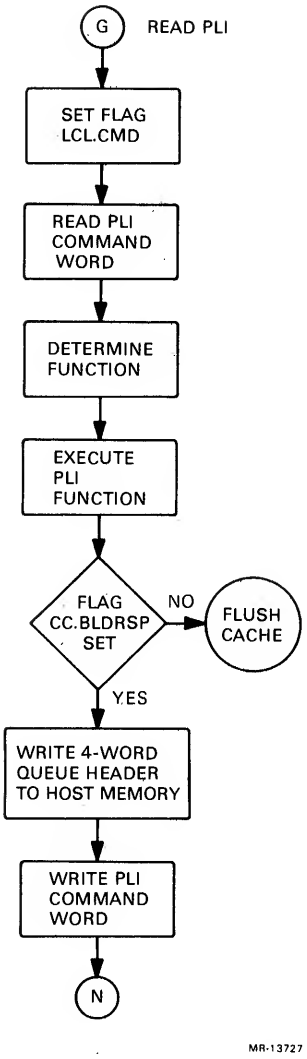


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 18 of 19 sheets)

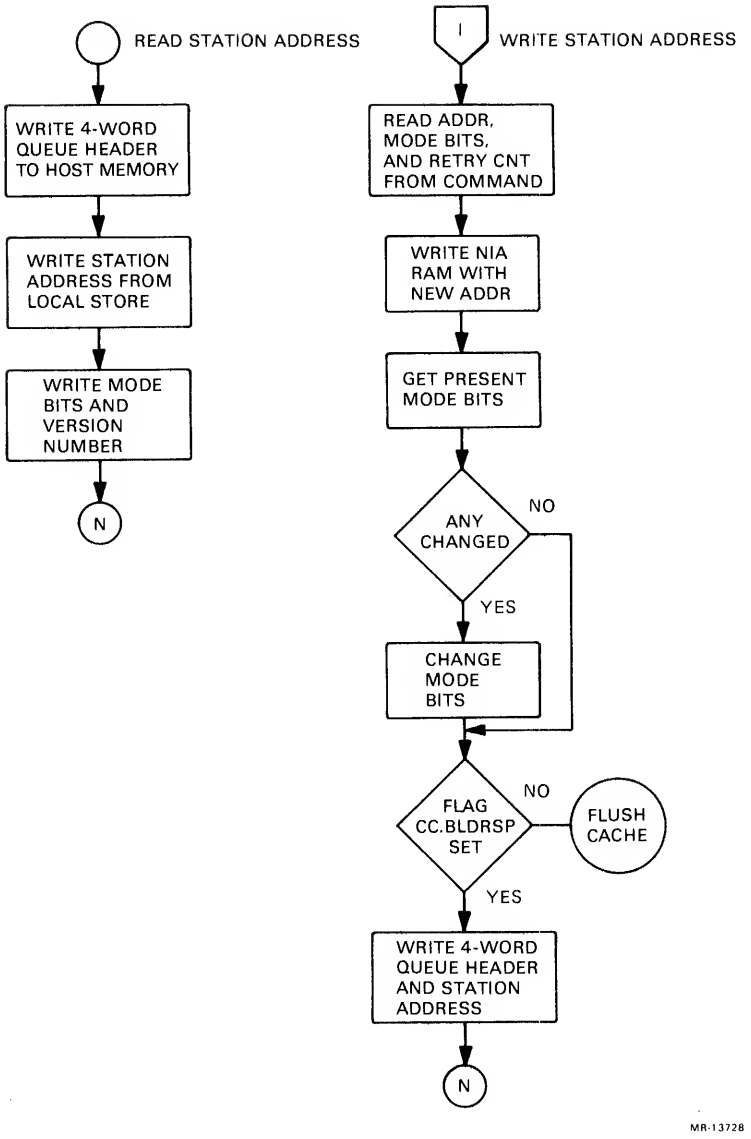


Figure 5-23 Transmit and Local Command Microcode Flow Diagram (Sheet 19 of 19 sheets)

Next, the microcode will set flag CC.BLDRSP, if the response bit is set in the operation code word and determine the operation code of the command.

This transmit microcode routine is then dispatched according to the operation code into nine subroutines:

1. Send datagram
2. Load multicast
3. Load protocol type table (PTT)
4. Read counters
5. Datagram receive
6. Write port logic interface (PLI)
7. Read PLI
8. Read station address
9. Write station address

1. Send Datagram Subroutine

Initially, the microcode in this subroutine looks at the PAD, the included cyclic redundancy check (ICRC), and the buffer segment descriptor (BSD) bits in the operation code word and sets the corresponding flag, if needed. The microcode then reads the length from the command and if the PAD flag is not set, a check is made to determine if the minimum frame size has been exceeded. If not, an error response is built and then linked onto the response queue. If the PAD flag is set, the code continues on to the next instruction.

If the ICRC bit is set in the FLAGS field, a 4-byte ICRC is added by the port driver to the datagram format, in the absence of a port-supplied CRC. This feature must be used in transmitting self-directed datagrams to maintain the CRC integrity of the frame. If the ICRC is not set, the port next examines the setting of the BSD bit.

Next, the microcode caches the protocol type, the free queue address, and the destination address. If the BSD flag is set, the code reads the BSD base address and then caches the BSD. In either case, the mover counts are cached and the cache state is set to NEW. If either flag XMIT.BUSY or XMIT.PEND is set, the cache is suspended and the port returns to the idle loop. When both flags are not set, the following occurs:

- Set flag XMIT.PEND
- Reset NIA transmit buffer address
- Write cached destination address to transmit buffer through data mover/formatter
- Write source address to transmit buffer through data mover/formatter
- Write 2-byte protocol type to transmit buffer.

The code now tests the PAD flag and if the flag is set, the actual length is written to the next 2 bytes in the transmit buffer. If not, the subroutine continues on to perform the following set of functions:

1. Use the smaller BSD/packet size for data mover
2. Divide smaller length by 4 for word count
3. Save full and partial word counts
4. Build channel command word (CCW) and start CBus
5. Load word count minus one to Am2910 loop count
6. Read word from host to formatter over CBus
7. Loop back for next word, if loop count is not zero
8. Loop back and shift, if full word (4 bytes)
9. Shift formatter output byte to PLI output
10. Move PLI output to transmit buffer.

When the loop count is zero, the microcode checks for any remaining partial words (1-3 bytes). If any partial word remains, the last word is read over the EBus to the mover/formatter and then shifted out to the transmit buffer.

The CBus is then stopped and the BSD flag tested. If it is a BSD, the new counts are calculated. When there are more bytes to transfer, a jump-back repeats the ten functions. If this is not a BSD or if the transfer is complete, the PAD flag is tested. With the PAD set, the remaining bytes are padded with zeros. The transmit end-of-frame (TX EOF) is written when padding is done or if the PAD is not set. Both flags XMIT.PEND and ICRC can now be cleared, but if they are set, the transmit CRC in the link control register is disabled.

With the flag XMIT.BUSY set, a transmit frame instruction to the NIA hardware is executed. The cache state is set to transmit and the port returned to the idle loop.

The assertion of transmit attention in the idle loop initiates a connection to the transmit cache and reads the transmit status. When the status is checked and errors are indicated, the corresponding error counter is incremented and an error status built. The following is a list of error types:

- Late collisions
- Transmit parity error
- Loss of carrier
- Transmit too long
- Heartbeat error

When there are transmit errors, the microcode restarts the retry subroutine.

Detection of a transmit buffer parity error causes the following to be performed:

- Clean the stack
- Clear flag XMIT.BUSY
- Mark cache as NEW.

Next, if retries are not exhausted, the microcode jumps back to recache the command, reloads the transmit buffer, and attempts to transmit again. If the retries are exhausted, the port is halted at location 7767.

If no errors or an error other than a transmit buffer PE is detected, the microcode checks for a collision retry and then increments the corresponding error counter. When more than 16 retries have been attempted, an error status is built and the time domain reflectometry (TDR) read.

In all error cases, the ICRC flag is tested and the disable CRC, if set, bit cleared in the link control register. The flag XMIT.BUSY is then cleared, and if no transmit error is detected, the running byte count is incremented and the transmitted frames counted.

If the response flag is not set when checked, the cache is marked free and the port returned to the idle loop. When the response flag is set, the completed operation code word, including the error status, is saved in cache and written into host memory. Next, the microcode attempts to link the response onto the queue. With an empty queue, the flag INTERRUPT.REQUEST is set. The cache is then marked as free and the port returned to the idle loop.

Next, from the idle loop, the microcode tests for a set interrupt request, which causes the port to wait for an idle EBus. When the EBus is idle, the response available queue bit in the CSR is set and a nonvectored interrupt generated. The interrupt request flag is cleared and the port returned to the idle loop.

2. Load Multicast Table Subroutine

The load multicast table subroutine is entered when the queue command operation code is a load multicast. This subroutine performs the following:

1. Sets the flag LOCAL.COMMAND
2. Builds a channel command word (CCW)
3. Starts the CBus
4. Reads the word from CBus to Am2901
5. Writes the local store (LS) with Am2901
6. Repeats steps 4 and 5 above 32 times (since there is no automatic increment of the LS for the loop).

This subroutine then tests the flag CC.BLDRSP. If the flag is set, a four-word queue header is written to the host memory and the flag LOCAL.COMMAND is cleared. The port then saves the complete operation code word in cache and tries to link onto the response queue. If the flag CC.BLDRSP is not set, the cache is flushed.

3. Load Protocol Type Table (PTT) Subroutine

The load protocol type table (PTT) subroutine is entered when the queue command operation code is a load PTT. This subroutine performs the following:

1. Sets the flag LOCAL.COMMAND
2. Builds a CCW
3. Starts the CBus
4. Reads word from CBus to the Am2901
5. Writes local storage with the Am2901
6. Repeats steps 4 and 5 above 47 times.

This subroutine then tests the flag CC.BLDRSP and if set, a four-word queue header is written to host memory and the flag LOCAL.COMMAND is cleared. The port then saves the complete operation code word in cache and tries to link onto the response queue. If the flag is not set, the cache is flushed.

4. Read Counters Subroutine

The read counters subroutine is entered when the queue command operation code is read counters. This subroutine performs the following:

1. Sets the flag LOCAL.COMMAND
2. Test CLR.CNT bit in operation code word and if set, causes the flag CLR.CNT to be set
3. Writes a four-word queue header to host memory
4. Builds a CCW
5. Starts the CBus
6. Reads a word from local store to the Am2901
7. Write host with the Am2901
8. Repeats steps 6 and 7 above 44 times.

Next, the assertion of the flag CLR.CNT clears the local store locations containing net counters. The flag LOCAL.COMMAND is then cleared. This subroutine saves the complete operation code word in cache and tries to link onto the response queue.

5. Data Scan Receive Subroutine

The datagram receive subroutine is entered when the queue command operation code is a datagram receive. This subroutine halts port operations at location 7750. A datagram receive is, as its designation implies, a receive function and not a transmit function, thereby indicating that fatal malfunction in the system has occurred.

6. Write Port Logic Interface (PLI) Subroutine

The write port logic interface (PLI) subroutine is entered when the queue command operation code is a write port. This subroutine performs the following:

1. Sets the flag LOCAL.COMMAND
2. Reads the PLI command word into the Am2901

3. Determines the PLI function
4. Executes the PLI function.

With the assertion of the flag CC.BLDRSP, a four-word queue header is written to the host memory -- in addition to the writing of a PLI command word. This subroutine saves the complete operation code word in cache and tries to link onto the response queue. If the flag is not set, the cache is flushed.

7. Read PLI Subroutine

The read PLI subroutine is entered when the queue command operation code is a read PLI. This subroutine performs the following:

1. Sets the flag LOCAL.COMMAND
2. Reads the PLI command word
3. Determines the PLI function
4. Executes the PLI function.

Next, the assertion of the flag CC.BLDRSP causes a four-word queue header to be written into the host memory -- in addition to writing the PLI command word. This subroutine saves the complete operation code word in cache and tries to link onto the response queue. If the flag is not set, the cache is flushed.

8. Read Station Address Subroutine

The read station address subroutine is entered when the queue command operation code is a read station address. This subroutine performs the following:

1. Writes four-word queue header into host memory
2. Writes the station address from local store to host memory
3. Writes mode bits and version number to host memory.

This subroutine saves the complete operation code word in cache and tries to link onto the response queue.

9. Write Station Address Subroutine

The write station address subroutine is entered when the queue command operation code is a write station address. The subroutine performs the following:

1. Reads address, mode bits, and retry count from the command
2. Writes the NIA physical address RAM with a new address
3. Changes any required mode bits.

If the flag CC.BLDRSP is set, a four-word queue header and station address is written into the host memory. This subroutine saves the complete operation code word in cache and tries to link onto the response queue. If the flag is not set, the cache is flushed.

APPENDIX A
INSTALLATION OF NIA20 IN KL10-D

A.1 OVERVIEW

This appendix describes the installation of the NIA20 network interconnect adapter in a KL10-D system. Figures A-1 and A-2 show the NIA20 installed in a KL10-D, rear and front views, respectively. Table A-1 itemizes the NIA20 parts and Table A-2 lists the harness and cable connections used in the NIA20/KL10-D installation.

The NIA20 installation uses assigned slots in RH20 logic assembly positions 4 and 5, with RH20 positions 6 and 7 reserved for installation of a CI20 computer interconnect. A system containing an NIA20 is limited to a maximum of four RH20s. In the installation of an NIA20, a module blank assembly, Digital P.N. 7019266-00, is used to prevent plugging any other module into RH20 position 4 (described in Section A.4.3, instruction 7 -- see Figure A-3).

NOTE

The prior or subsequent installation of a CI20 computer interconnect with an NIA20 requires minor deviations from the following procedures, which are described herein when applicable.

Installation of the NIA20 in an existing system requires implementing the following procedures:

1. Unpacking and checkout of installation kit
2. Preinstallation checkout
3. Backplane wire adds
4. Installation of port modules
5. Installation of power supply regulator
6. Installation of NIA card cage
7. Installation of NIA current limiter
8. Installation of dc power harness
9. Installation of vane switch harness
10. Installation of dc voltage monitor harness and module
11. Installation of PLI bus
12. Installation of fan ac cable and power cord
13. Installation of internal NIA cable
14. Installation of KL10 adapter board and blank module assembly
15. Checkout.

The following sections provide detailed instructions for performing each of these above installation procedures.

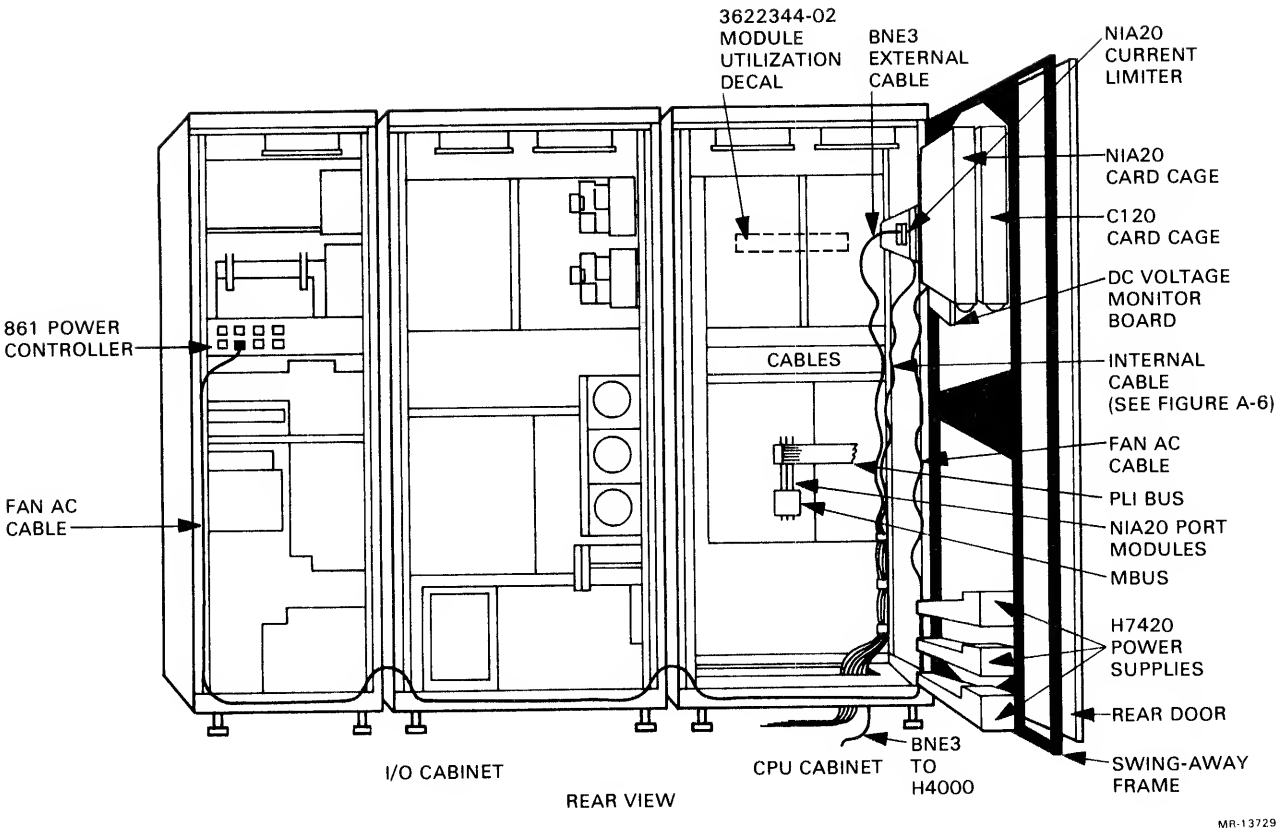


Figure A-1 NIA20 in KL10-D, Rear View

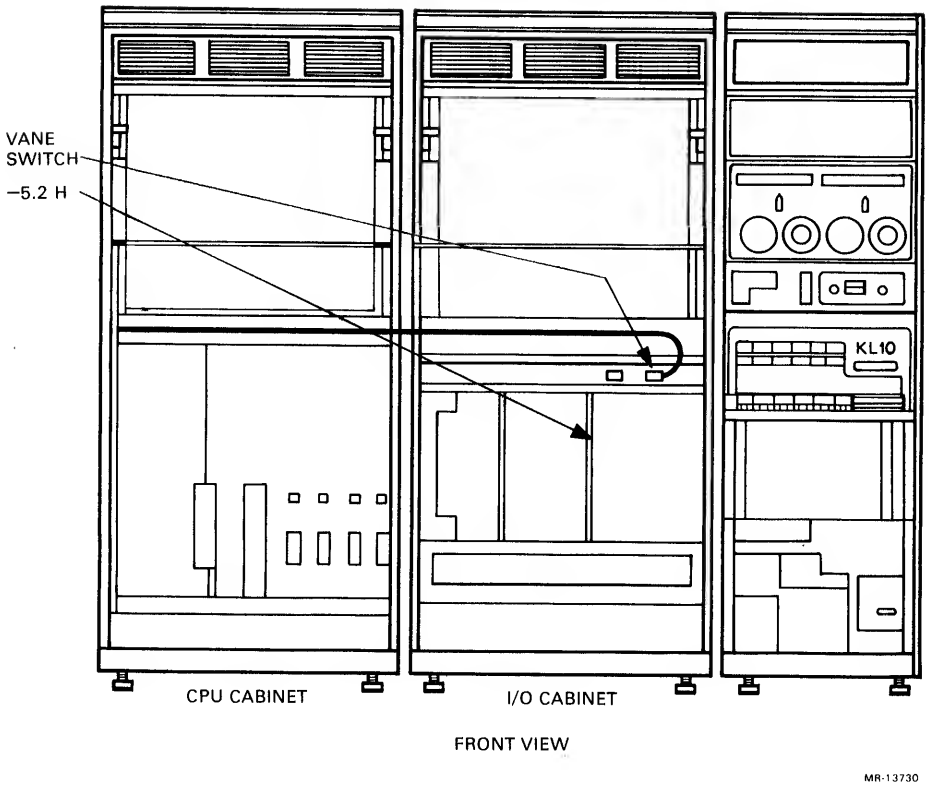


Figure A-2 NIA20 in KL10-D, Front View

Table A-1 NIA20 in KL10-D, Parts List

Line Item	Part No.	Description	Qty
1	7019268-00	Card cage assy IPA-20-L	1
2	7019268-01	Card cage assy CI20	1
3	7428312-01	Bracket, interface	1
4	7430279-01	Bracket, support	2
5	9007786-00	Retainer, U-nut 10-32X	9
6	9006073-01	Screw, mach pan phil 10-	17
7	9107240-09	Wrap, cable, .250 OD vinyl wht	A/R
8	9105740-55	Wire (wrap) 30 AWG KYNAR UL14	A/R
9	1213716-00	Spacer, foam polyu 1/2	5
10	7020539-06	Cable, fan ac	1
11	7019274-06	Cable, fan ac	1
12	7021197-01	Harness, dc-5.2 dc+5	1
13	7430277-01	Support, bracket interface	1
14	7019893-3L	Cable assy Ethernet	1
15	BC06R-08	BC06R I/O cable	1
16	7019266-00	Module blank assy	1
17	M3002-00	CI20 microprocessor, multiwire HE	1
18	M3003-00	CI20 CBus/PLI interface, multiwire	1
19	M3001-00	CI20 EBus interface, multiwire H	1
20	9007032-00	Tie, cable bundl. Dia 0-1-3/4"=101	A/R
21	1213715-00	Clip, flat cable w/adhesive bk	5
22	H7440-00	POA1 H7440	1
23	L0072-00	NI20 (KL10 to NI adapter)	1
24	7014103-00	Blank module assy	1
25	9107673-06	Pwr cord, term 3-14 SJT 115	1
26	7011432-02	Pow cord extension 50 Hz	1
27	9007651-00	Washer, lock external steel	17
28	9006664-00	Washer, flat SST	17
29	7021196-01	Harness, dc voltage monitor	1
30	7021194-01	Harness, vane switch	1
31	5414506-01	Voltage, monitor board	1
32	7019270-1J	Bus, cable, M assy	1
33	3621499-01	Label, DCV monitor CI20	1
34	3613272-00	Label, adh back, Mylar cap	1
35	9007031-00	Tie, cable bundl. Dia 0-3/4"=101	36
36	9008264-00	Mount, cable tie, adhesive back	A/R
37	3621498-01	Label, airflow CPU CI20	1
38	5415695-01	Current limiter	1
39	9006022-01	Screw, mach pan Phil 6-	4
40	9006633-00	Washer, lock internal steel	4
41	7020488-00	Cable, short switch vane	1
42	3617674-00	Label, serial/power W/O UL + CSA	1
43	3617674-01	Label, serial/power W UL & CSA	1
44	3617880-09	Label, class A subassembly	1
45	3621501-02	Label, module location, NI20	1

Table A-2 NIA20 in KL10-D, Harness and Cable Connections

Harness Connections

Parts List Item No.	Harness Terminals Point	Connection	Connection	Remarks
12	--	P1	NIA20 BP J2	--
	-	P3	NIA20 BP J1	--
	5	--	CPU #3 BP GND	--
	6	--	CPU #3 BP -5.2 H	--
	--	P2	J1	
13	--	P1	H7440 J1	See Figure A-5
	7	--		
	8	--		
30	--	P1	Fan Bracket J1	See Note 1
	--	P2	See Figure A-11	See Note 1
	--	J1	See Note 2	See Note 1
	--	P3	NIA20 BP J6	See Notes 1 and 3
29	--	P2	NIA20 BP J5	
	--	P1	Mon. Board J1	Parts List Item 31
	6	--	+5V Monitor Board J1-5	See Figure A-2
41	P3		CI20 Cable Vane Switch	See Note 4
	P2	--	NIA20 Fan Bracket J1	See Note 4
	P1	--	CI20 Fan Bracket J1	See Note 4

NOTES:

1. Items not needed when CI kit is installed:

Item	Qty	Part No.	Description
5	8	9007786-00	Retainer, U-nut 10-32X
6	8	9006073-01	Screw, Mach Pan Phil 10-
27	8	9007651-00	Washer, lock ext ST
28	8	9006668-00	Washer, flat SST
3	1	7428312-01	Bracket, interface
30	1	7019862-00	Harness, vane switch
4	2	7430279-01	Bracket, support
13	1	7430277-01	Support, bracket interface
7	A/R	9107240-09	Wrap, cable, .250 OD vinyl Wht

2. J1 on parts list Item 30 (harness, vane switch) connects with existing connector P4 (see Figure A-12).

3. Relocate cable from CI20 card cage at J6 connector and insert into NIA20 card cage J6 connector.

Table A-2 NIA20 in KL10-D, Harness and Cable Connections (Cont)

4. Parts list Item 4 used when CI20 and NIA20 are installed together.

Cable Connections

Parts List Item No.	From Unit	To Location	Ref. Desig.	Unit	Location	Ref.Desig.	Remarks
10 or 11	C.Cage Fan Brk	Gnd J2	Gnd P2	25 or 26	J1	P1	NIA20-CB
14	C.Cage	J4	P1	41	P2		
25/26	Item 10 or 11	P1	J1	861 PC	P1		Connect to any avail. switched outlet
1/2	C.Cage	Gnd	-	Cabrail	Hole 1	-	
15	NI20 BP	J3	PI Stripe Down	RH20 DTE CC	M3003 J2	P2 Stripe Up	
32	M3003 M3001	J1 J1	P3 P1	M3002	J1	P2	
38	RH20 BP	B10N1	-	RH20 BP	B13B1	-	
		B13B1	-		B19B1	-	
		C10L2	-		B13B2	-	
		B13B2	-		B19B2	-	
		C10K2	-		B13U1	-	
		B13U1	-		B19U1	-	
		C10T2	-		C13B1	-	
		C13B1	-		C19B1	-	
		C12H2	-		C13N1	-	
		C13N1	-		C19N1	-	
		C12L1	-		C13B2	-	
		C13B2	-		C19B2	-	
		C14H2	-		A15R2	-	
		C14F1	-		F15A1	-	
		A14J2	-		A15E1	-	
		C14P1	-		A15D2	-	
		C14K2	-		A15S2	-	
		B14J1	-		B15A1	-	
		C20H2	-		A21R2	-	
		C20F1	-		F21A1	-	
		A20J2	-		A21E1	-	
		C20P1	-		A21D2	-	
		C20K2	-		A21S2	-	
		B20J1	-		B21A1	-	

A.2 UNPACKING AND CHECKOUT

Before unpacking any equipment, move all boxes into the computer area. Check the shipment against the packing list to be sure that all boxes were sent. If any boxes are missing, contact the customer and the branch field service manager. Check that all boxes are sealed, and there is no sign of external damage, such as dents, holes, or damaged corners.

If any boxes are open or damaged, document it on the installation or field service report and inform the customer. Open the boxes one at a time, starting with the box marked "READ ME FIRST" and find the packing slip. Check the contents of the box against the packing slip and examine each item for damage. Note missing or damaged items on the installation report or field service report.

This completes the unpacking and checkout phase. Advise the branch field service manager of any problems during this phase. If any items are damaged, the branch field service manager may want the customer to file an insurance claim. For missing items, the branch field service manager should get a short-ship request.

A.3 EQUIPMENT NEEDED FOR INSTALLATION AND CHECKOUT

The following equipment is required for installation and checkout of the NIA20:

1. Wire wrap tool, No. 30 AWG, Digital P.N. 29-18301
2. Wire unwrapping tool, No. 30 AWG, Digital P.N. 29-13513
3. Regular Phillips screwdriver
4. Tektronix 475 oscilloscope or equivalent (100 MHz)
5. KLAD pack
6. Scope, digital voltmeter.

A.4 INSTALLATION PROCEDURE

A.4.1 Preinstallation Checkout

Before performing the installation, verify that the currently configured system is operating properly, to preclude the possibility of present system problems being ascribed to the NIA20 after its installation.

1. Remove all customer media to minimize the possibility of corrupting customer data.

2. Mount the KLAD pack, bring up the diagnostic monitor, and run the "B" string to verify that the system is working properly.

3. Power-down the system.

4. Verify that the system has a M8532-YA board installed. If not, replace the currently installed M8532 with a M8532-YA.

5. RH20 positions 4 and 5 are used for the NIA20. If there is an RH20 in position 4, remove it. If there is an RH20 in position 5, leave it temporarily installed and perform diagnostic DFRHB to verify the reliability of the backplane wiring. If there is no RH20 in position 5, relocate a module from one of the other RH20 positions to position 5.

6. Power-up the system and run diagnostic DFRHB. This verifies that the backplane wiring of RH20 position 5 is functional. Power-down the system and reinstall the RH20 in its original position.

7. Perform diagnostic DFRHB also in RH20 position 7 to verify the reliability of existing backplane wiring in RH20 positions 6 and 7, before implementing any NIA20 modifications.

8. Power-down the system and reinstall the RH20 in its original position.

A.4.2 Backplane Wire Adds

For the NIA20 installation, 24 new wires must be added to RH20 backplane positions 4 and 5. An examination of the RH20 backplane must be performed to confirm the physical addition of the wire wraps listed in Table A-3. The table contains a Check column for the wire installer to record installation progress.

To prepare the wire adds, strip approximately 1 inch of insulation from the wire to allow sufficient turns to be made on the wire wrap post. After each wire is added, enter a check in the blank space adjacent to the wire listing in Table A-3.

To assure the reliability of the new wiring, an ohmmeter check of each new wire add should be performed by a person other than the wire installer.

Table A-3 NIA20 in KL10-D Wire Adds

Signal Name	From	To/From	To	Check
EBUS D11 L	B10N1	B13B1	B19B1	_____
EBUS D12 L	C10L2	B13B2	B19B2	_____
EBUS D13 L	C10K2	B13U1	B19U1	_____
EBUS PARITY L	C10T2	C13B1	C19B1	_____
EBUS PI00 L	C12H2	C13N1	C19N1	_____
EBUS PARITY ACTIVE L	C12L1	C13B2	C19B2	_____
MPR7 MWBUSCTLFLD01 H	C14H2	A15R2		_____
MPR7 MWMGCFLD08 H	C14F1	F15A1		_____
MPR7 MWTIMEFLD H	A14J2	A15E1		_____
CBI1 CLK2 L	C14P1	A15D2		_____

Table A-3 NIA20 in KL10-D Wire Adds (Cont)

Signal Name	From	To/From	To	Check
CBI2 CLK4 L	C14K2	A15S2		
CBl2 CCCHANERR L	B14J1	B15A1		———
MPR7 MWBUSCTLFLD01 H	C20H2	A21R2		———
MPR7 MWMGCFLD08 H	C20F1	F21A1		———
MPR7 MWTIMEFLD H	A20J2	A21E1		———
CBI1 CLK2 L	C20P1	A21D2		———
CBI2 CLK4 L	C20K2	A21S2		———
CBI2 CCCHANERR L	B20J1	B21A1		———

A.4.3 Installation of Port Modules

Protective backing is placed on the lower third noncomponent side of each port module and the upper third of the noncomponent side of the M3002. As the modules are inserted and/or removed, the protective backing protects the MBus and PLI bus cables. The protective backing should not interfere with the card guide or cover the gold finger contacts on the module. Insert the port modules as follows:

1. Connect MBus cable, Digital P.N. 7019270-1J. Be sure to orient the cable so that the flat wire comes out of the cable header away from the board, as shown in Figure A-3.

2. Insert the M3001 EBus interface/port ALU module in the rightmost slot of RH20 position number 5 (slot 19, looking at the backplane from the module side). The arrow on cable should be aligned with the arrow on board connector.

3. Connect the MBus cable to the M3002 port microprocessor module as shown in Figure A-3.

4. Insert the module M3002 in slot 20 to the left of the installed M3001 as shown in Figure A-3.

5. Connect the MBus cable to the M3003 module as shown in Figure A-3.

6. Install the M3003 CBus/PLI interface module in slot 21, which is located to the left of the installed M3002.

7. Install the module blank assembly, Digital P.N. 7019266-00, in RH20 position 4, slot 22. This assembly blocks slots 22, 23, and 24. It prevents modules from being inserted into RH20 position 4 and provides a baffle for system cabinet airflow.

8. Perform an ohmmeter check between PT17U and ground to verify that there are no shorts to ground.

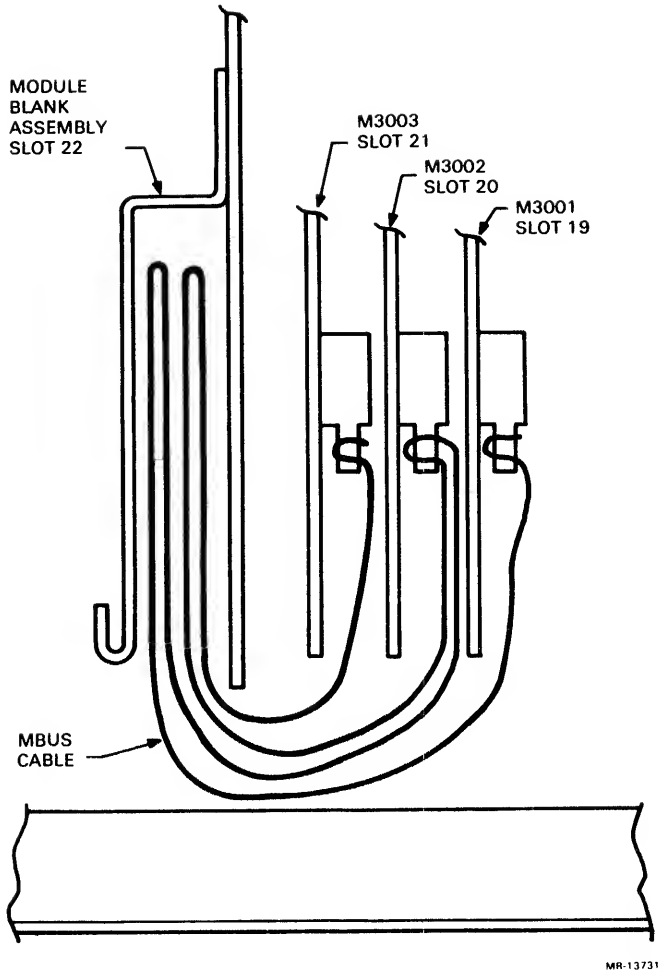


Figure A-3 MBus Cable Interboard Connection, Top View

9. Fold the MBus cable into the module blank assembly as shown in Figure A-3.

10. Close the module door.

11. Attach the self-sticking module utilization decal, Digital P.N. 3622344-02, on the upper rear baffle panel.

12. Power-up the KL10.

13. Readjust the existing +5 V power supply to 5.0 +/- 0.25 V. This adjustment is located on H7420 number 1 in H744 number 4. The location of this regulator is farthest away from the circuit breaker. The voltage is monitored at +5F, between PT17U and ground.

14. Type MR (CR) with KLDCP loaded and running; then type FX1 (CR) in response to the command prompt, as shown below:

    >. MR (CR)

    >. FX1 (CR)

15. De-skew the port modules using a Tektronix 475 or equivalent 100 MHz minimum oscilloscope and perform the following steps. Figures A-4 and A-5 shows the acceptable waveform pattern displays of the NIA20 de-skew timing.

16. Connect channel 1 of the oscilloscope to MTR MBOX CLK H, 4D33P1, on the CPU backplane. Use a ground clip.

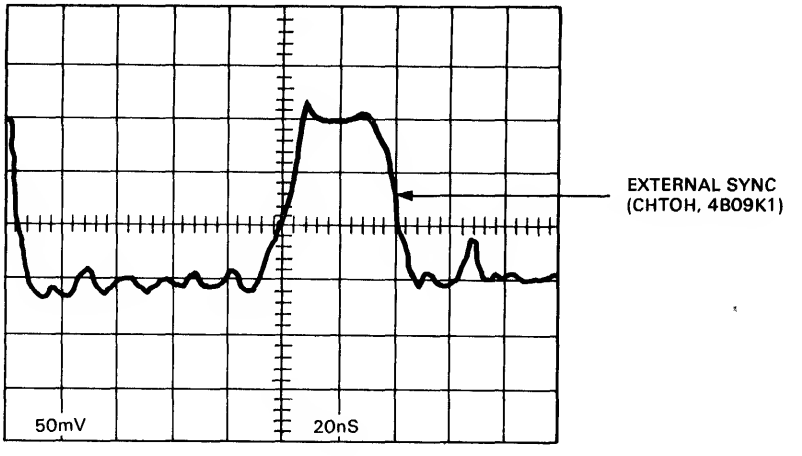


Figure A-4 NIA De-skew Timing. External Sync (CHT0 H)

17. Set the time base to 20 ns.

18. Set channel 1 vertical gain to 0.5 V/division. Set the ground reference to 1.3 V above the horizontal center level of the oscilloscope (MTR MBOX CLK H is an ECL signal).

19. Set the oscilloscope sync to positive external.

20. Connect external sync input to CHT0 H, 4B09K1 on the CPU backplane. Use a ground clip.

21. Connect channel 2 to CDS1, EBUS CLK L, 2A21F1 on the I/O backplane. Set the channel 2 vertical gain to 0.5 V/division. Use a ground clip. To measure TTL voltages,

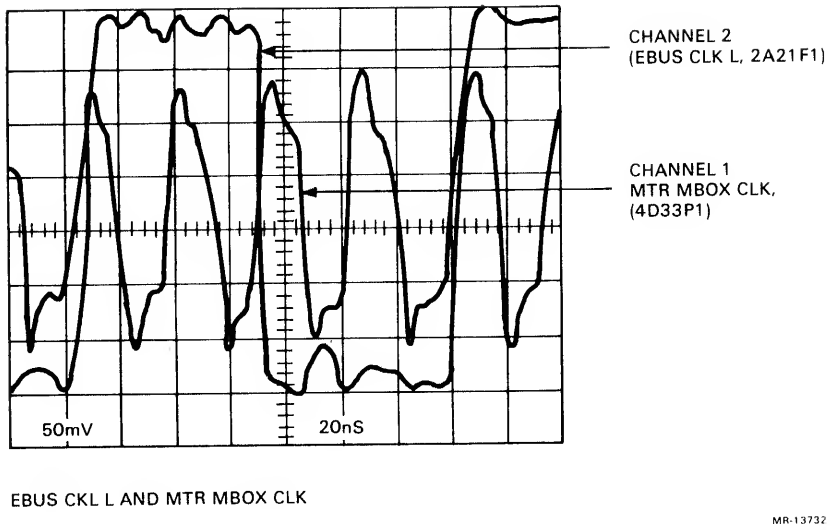


Figure A-5 NIA20 De-skew Timing. EBus CLK L and MTR MBOX CLK

set the ground reference to 1.5 V below the horizontal center line of the oscilloscope.

22. Press the trigger view switch of the oscilloscope and display the external sync. Adjust the display, so that the rising edge of the external sync aligns with the vertical center line of the oscilloscope.

23. Display MBOX CLK H, channel 1. Identify the rising edge of MBOX CLK H that occurs prior to the vertical center line of the oscilloscope. Display channel 1 and channel 2.

24. Put the KL10-D in the override fault state. Remove the I/O rear door to access the I/O backplane.

25. Locate the bottom potentiometer on the clock module (M8559) in slot 12 of the I/O backplane. Using this potentiometer, adjust the falling edge of channel 2, EBUS CLK L so that it crosses the rising edge of MBOX CLK H. This crossing occurs on the horizontal center line of the oscilloscope.

26. Disconnect all probes.

27. Mount the KLAD pack on the front end RP06.

28. Load and run diagnostic DFPTA to verify proper functioning of the port modules. If the modules fail, troubleshoot as directed by the diagnostic. If the modules are functioning properly, continue with the installation.

A.4.4 Power Supply Regulator Installation

Three H7420 power supplies are located on the rear swing-away frame door of the I/O cabinet as shown in Figure A-1. The H7440 regulator to be added is installed in the upper H7420 power supply location. This additional +5 V regulator is required to support the NIA20 card cage and is installed as follows (see Figure A-6):

1. Remove the spare slot filler panel from slot 5 of the H7420 number 1 power supply. Save all existing hardware.

2. Take the new H7440 regulator from the kit and install the H7440 in slot 5 of H7420 number 1, using two screws on top and one thumbscrew at the bottom. Some systems may use H744 or H7440 regulators.

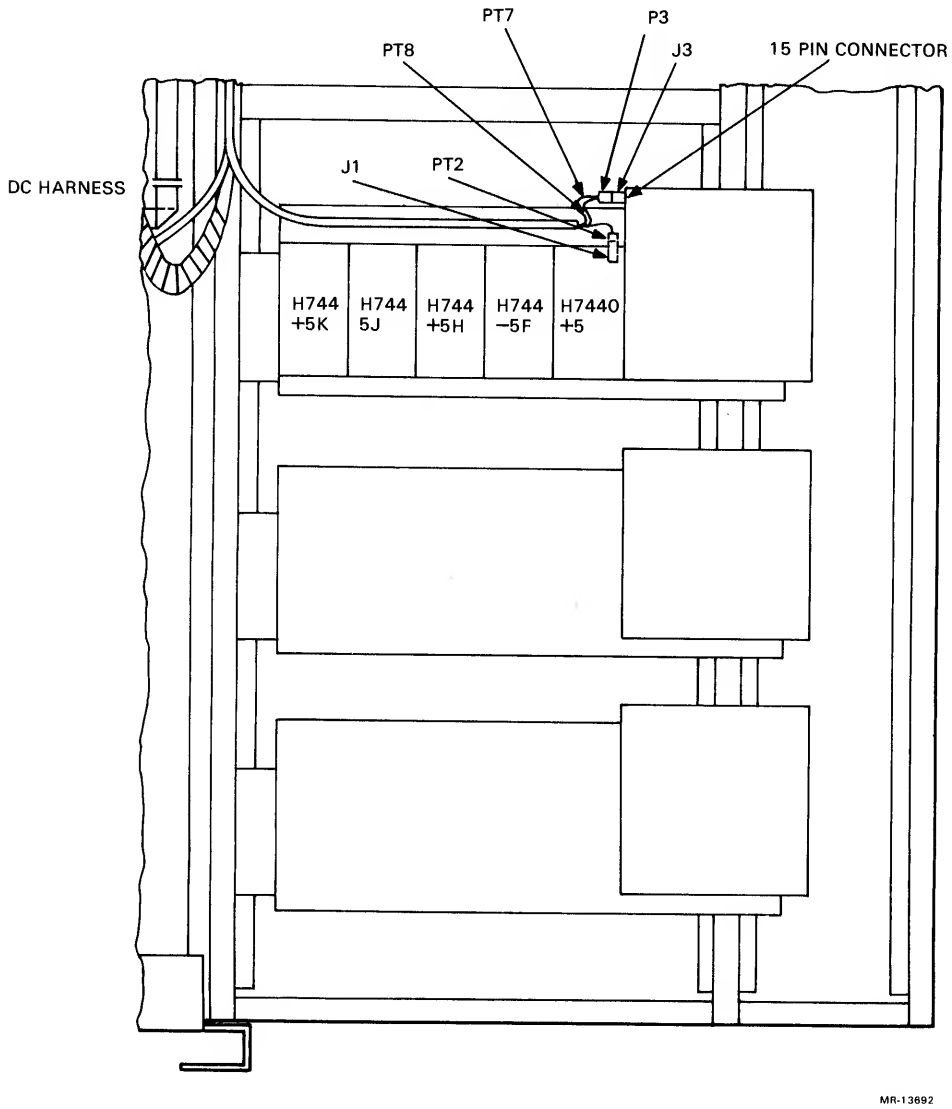


Figure A-6 H7420 Power Supply

A.4.5 Installation of NIA20 Card Cage/Cable

To install the NIA20 card cage and the internal NIA20 cable, perform the following operations (see Figures A-7 and A-8):

1. Install the two NIA20 mounting brackets, Digital P.N. 7430278-01, as follows:

   NOTE

   When a CI20 is installed, the NIA20 is mounted as shown in Figure A-1.

   Remove and reposition any tie-wrapped cables from the right frame member of the CPU cabinet (viewed from the rear) to accommodate the NIA20 mounting brackets and card cage.

   Install a total of 8 U-nuts (Tinnerman nuts), Digital P.N. 9007786-00, in the swing-away frame of the CPU cabinet (viewed from the rear).

   Insert Tinnerman nuts into frame holes 3, 4, 41, and 42 counting down from the top of the swing-away frame (see Figure A-7). Four Tinnerman nuts are inserted into each side of the frame.

   Check that the lip of each support bracket is at the bottom. Use four 10/32 one-half inch Phillips panhead machine screws, Digital P.N. 9006073-01 and four No. 10 starlock washers, Digital P.N. 9007651-00 on each side of the swing-away frame.

2. Locate the NIA20 internal cable strain relief (white) on the left side of NIA20 card cage (see Figure A-8). Because of the inaccessible location of this strain relief once the card cage is installed, the internal cable is routed through it and connected to the rear J4 connector on the card cage prior to installing the card cage.

3. Locate the three-foot internal NIA20 cable, Digital P.N. 7019893-3L.

4. Prepare the internal NIA20 cable for installation by positioning a stick mount on the right-side vertical frame member of the CPU cabinet (viewed from the rear), above the bracket interface (see Figure A-9).

5. Route the three-foot internal NIA20 cable through the white plastic strain relief on the NIA20 card cage.

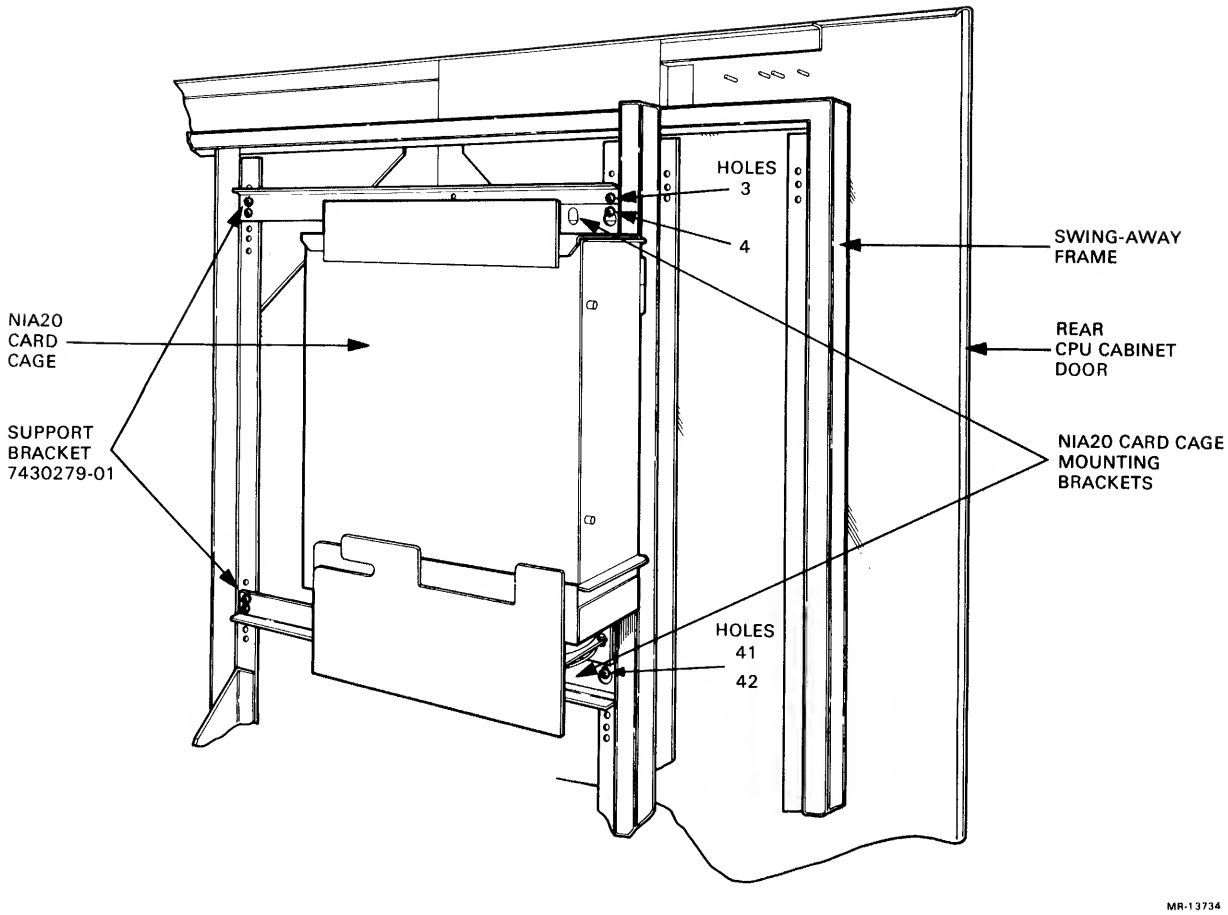


Figure A-7 NIA20 Card Cage in KL10-D

6. Connect the internal NIA20 cable to the J4 connector (see Figure A-8) on the NIA20 card cage and route the cable as shown in Figure A-1. The cable connector engages a detent when properly seated.

7. Mount the NIA20 card cage on the two NIA20 mounting brackets that overlay the support brackets, Digital P.N. 7430279-01 (see Figure A-7), using eight 10/32 screws, external lockwashers, and flat washers. Hang the NIA20 card cage on the top four screws, and then install the bottom four screws.

8. Install the NIA20 card cage ground cable (see Figure A-8) as follows:

   Install a Tinnerman nut in hole 1 of the right side frame member in the CPU cabinet (viewed from the rear).

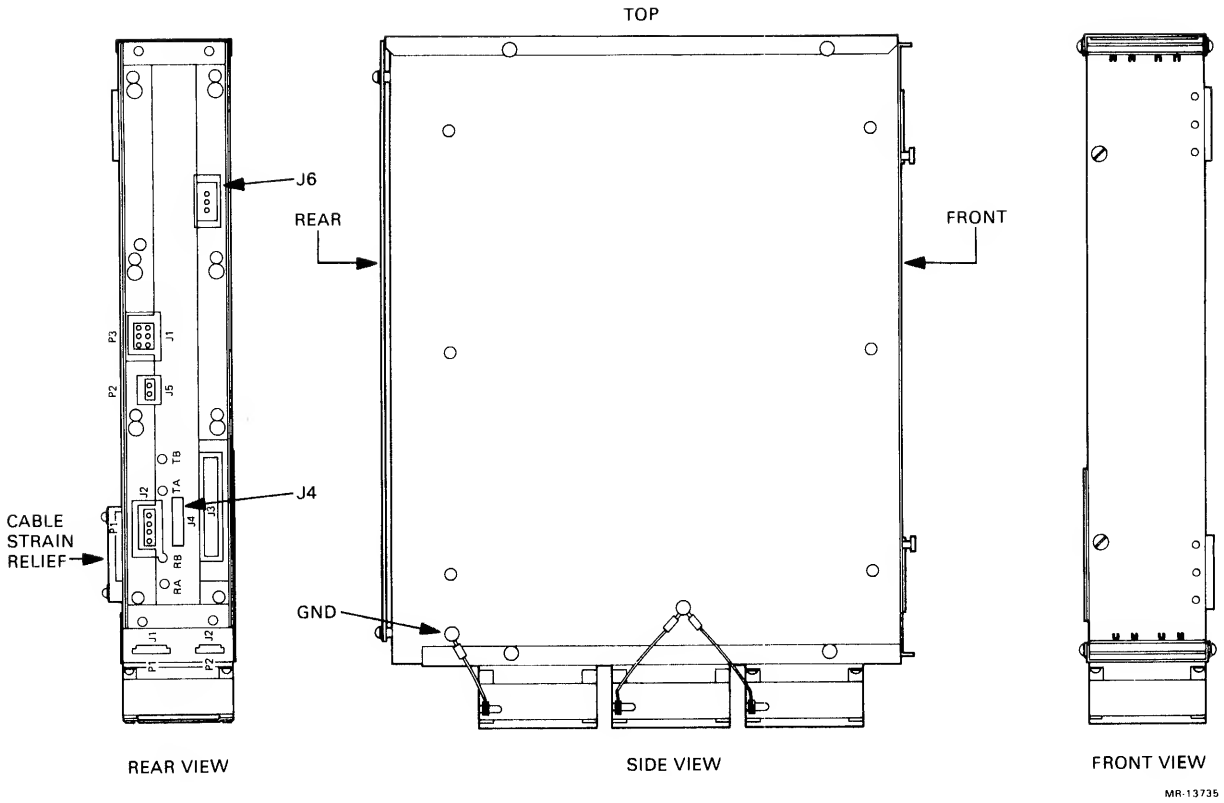


Figure A-8 NIA20 Card Cage Views

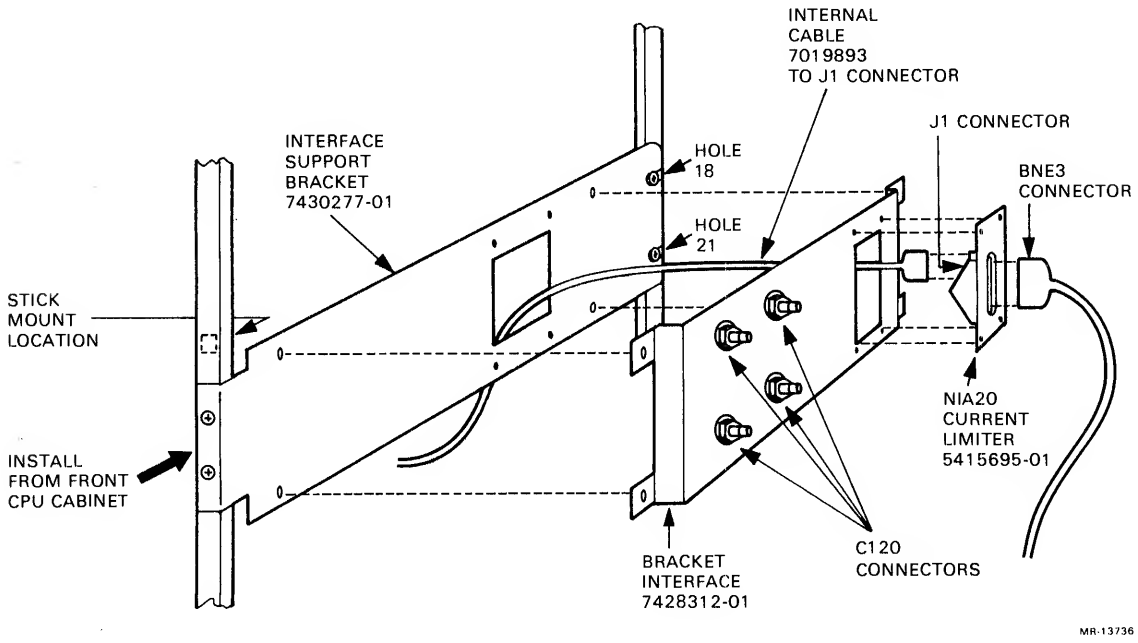


Figure A-9 NIA20 Current Limiter

Connect the ground cable on the right side frame member by inserting a screw and using a starwasher on each side of the ground cable.

Attach a ground label, Digital P.N. 3613272-00, closest to hole 1.

9. Run the internal NIA20 cable to the previously positioned stick mount and insert its other end into the rear J1 connector of the NIA20 current limiter.

A.4.5.1 Installation of NIA20 Current Limiter -- The NIA20 current limiter, Digital P.N. 5415695-01 is preinstalled on the bracket interface, Digital P.N. 7428312-01, as shown in Figure A-9. Install the bracket interface on the interface support bracket and connect the internal and external cables in the current limiter as follows:

1. Install the interface support bracket, Digital P.N. 7430277-01, in holes 18 or 21 located on the right side of the CPU cabinet (rear view) using four 10-32 screws, external lockwashers, and flat washers.

2. Connect the internal cable to the rear J1 connector on the NIA20 current limiter (see Figure A-9).

3. Connect the BNE3 external cable to the front P1 connector on the NIA20 current limiter. The other end of the external cable connects to the Ethernet transceiver.

A.4.5.2 Harness Installation -- The following harnesses are to be installed:

1. DC power harness
2. Vane switch cable
3. DC voltage monitor cable
4. Fan ac cable and power cord
5. PLI bus
6. BNE3 external cable.

Figure A-10 shows a diagram of the harness and cable interconnections. The harnesses are installed as follows:

1. Install tie-wraps approximately eight inches apart on all harnesses. When routing cables close to internal assemblies, use spiral wire-wrap to protect the cables.

2. Locate the dc power harness, Digital P.N. 7021197-01 (see Figure A-11), and the black and blue wires labeled PT5 and PT6. Connect the black wire to -5.2 ground and the blue wire to -5.2F in the I/O cabinet (see Figure A-2).

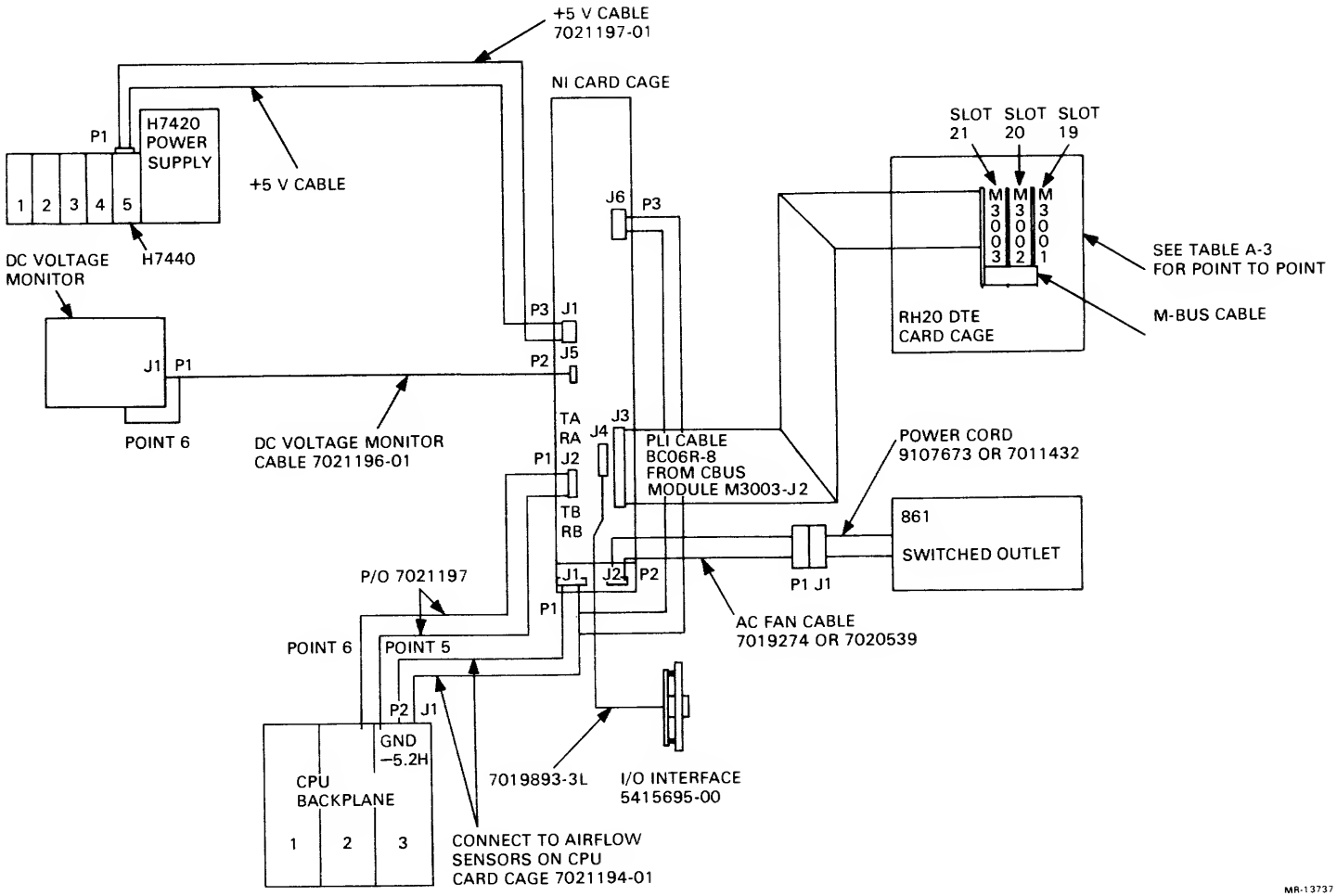


Figure A-10 NIA20 Harness and Cable Interconnection Diagram

3. Locate and connect P1 of the dc power cable into connector J1 of the NIA20 card cage backplane (see Figure A-8). Next, connect P3 of of DC power cable into J2 of the NIA20 card cage.

4. Tie-wrap the new harness to existing KL10-D I/O power harnesses and route this cable through the cabinet floor as shown in Figure A-1. Use spiral wrap along the harness where it contacts the side of the CPU frame member nearest the H7420 power supplies (see Figure A-1).

5. Locate the red and white wires labeled PT7 and PT8 at the other end of DC power cable (see Figure A-11). Disconnect P3 from power supply H7420 number 1. Then connect PT7 and PT8 to pins 3 and 4, respectively, on P3 of the H7420. Then reconnect P3 to the H7420.

6. Connect P2 of the harness to connector J1 of the H7440 regulator (see Figure A-6).

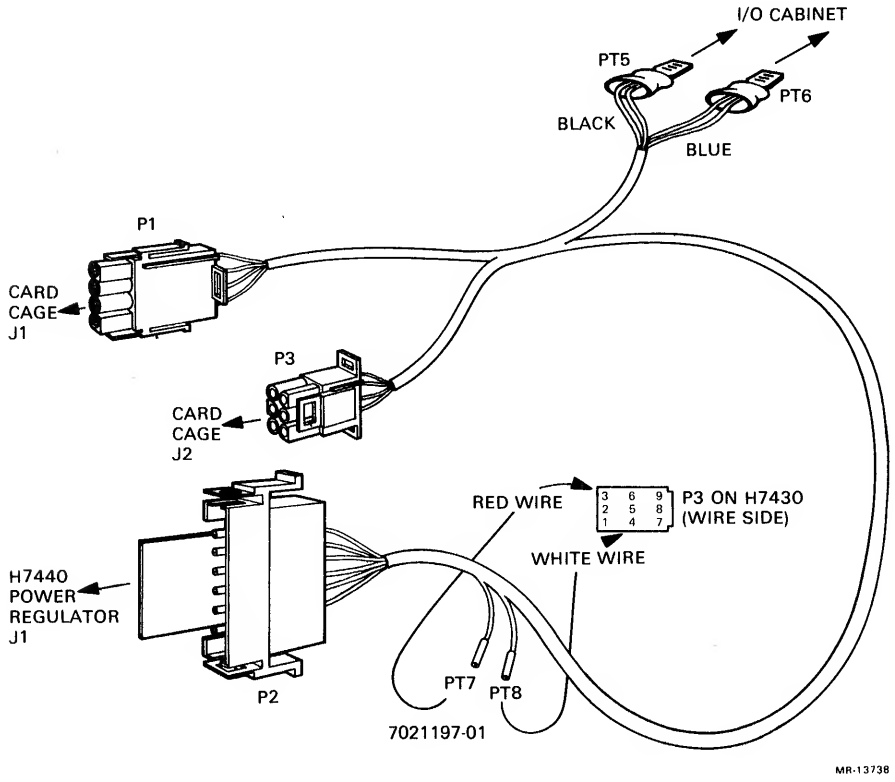


Figure A-11 DC Power Cable

7. Locate the vane switch cable, Digital P.N. 7021194-01 (see Figure A-12). Connect P1 of the vane switch cable to connector J1 on the NIA20 card cage (see Figure A-8).

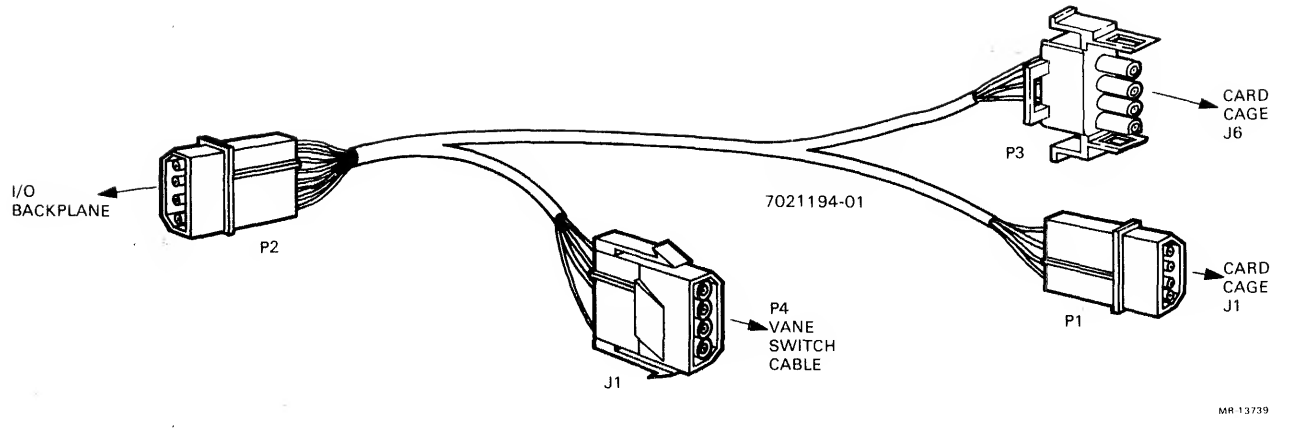


Figure A-12 Vane Switch Cable

Also, connect P3 of the vane switch cable to connector J6 on the NIA20 card cage. Use stick mounts and spiral wire wrap as needed to route and protect the vane switch cable.

NOTE

When a CI20 is installed, the short switch vane cable, Digital P.N. 7020488-00, is used in a combined CI20/NIA20 installation. Consult the CI20 reference manual (Digital order number EK-CI20-RM-001) for other applicable CI20 installation procedures.

8. Remove the original KL10-D CPU vane switch cable (P4) and connect this to the NIA20 vane switch cable J1 (see Figure A-13).

9. Connect P2 of the vane switch cable to the original KL10-D CPU vane switch assembly in the I/O cabinet.

10. Insert the other end of the vane switch cable as shown in Figure A-2.

11. Apply the CPU/NIA20 air flow fault decal over the existing CPU air fault message decal on the 863 fault switch.

12. Locate the dc voltage monitor cable, Digital P.N. 7021196-01 (see Figure A-14).

13. Connect P2 of the dc voltage monitor cable to connector J5 on the NIA20 card cage backplane (see Figure A-8) and connect the P1 end of the cable to the dc voltage monitor

board, Digital P.N. 5414506-01, mounted on the front of the swing-away frame located inside the CPU cabinet (see Figure A-1).

14. Locate the switches on the dc voltage monitor board. Only switch S1 should be on, while all other dc voltage monitor board switches should be off.

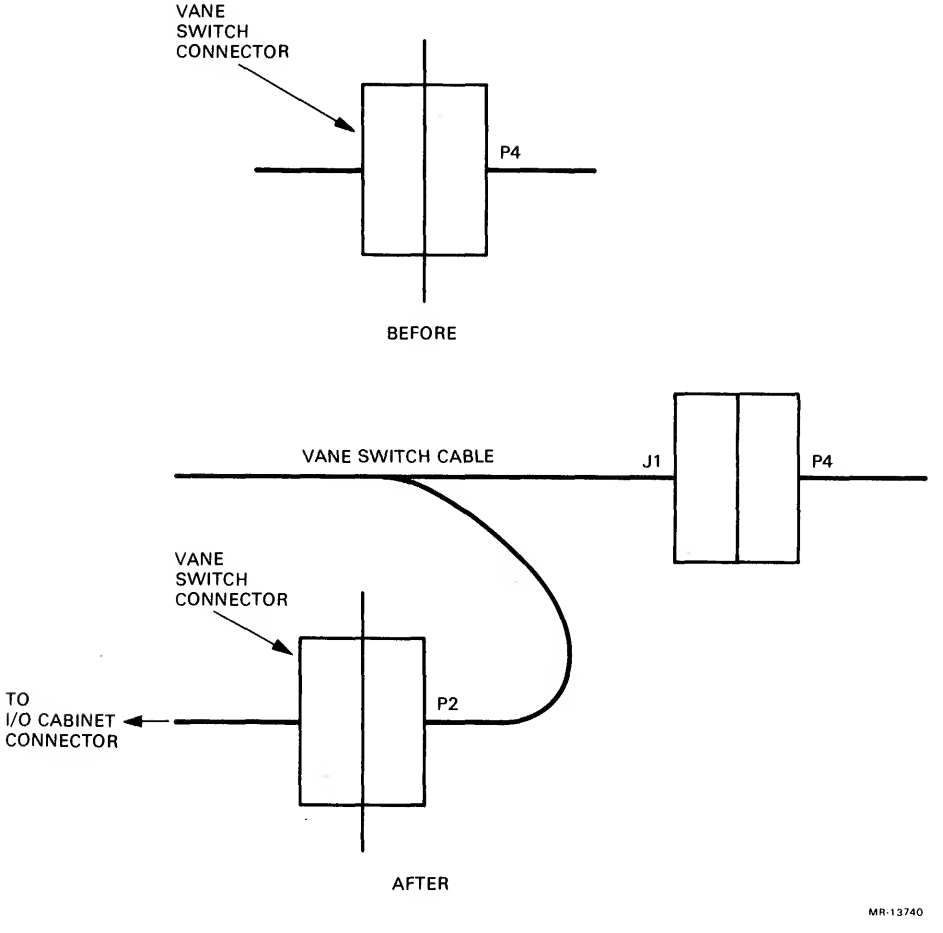


Figure A-13 Vane Switch Harness Installation

15. Insert the dc voltage monitor board into the +5 V slot of the dc voltage monitor panel.

16. Attach the monitor panel decal, Digital P.N. 3621501-02, to indicate the slot used for the NIA20 dc voltage monitor board.

17. Connect the remaining single orange wire of the dc voltage monitor cable to a location adjacent to the existing orange wire on the dc voltage monitor board zone +5L.

18. Tie-wrap the dc voltage monitor and vane switch harnesses to the dc power cable. Use adhesive-backed square cable mounts to support the harness on the baffle door and cable ties and wire wrap along the cabinet frame.

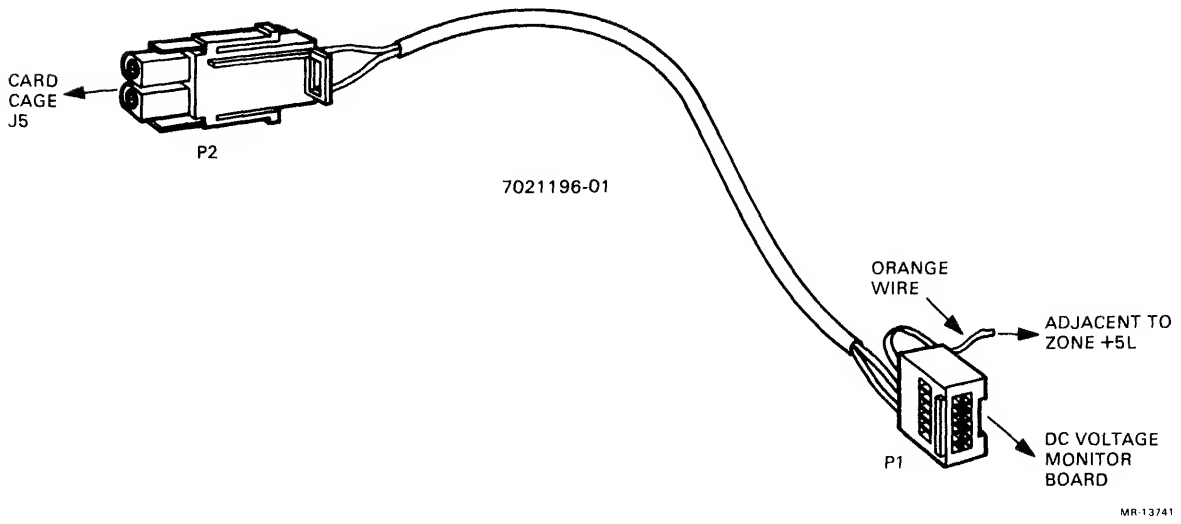


Figure A-14 DC Voltage Monitor Cable

19. Locate the fan ac cable, Digital P.N. 7019274-06 (120 Vac 60 Hz) or 7020539-06 (240 Vac 50 Hz), and the power cord, Digital P.N. 9107673-06 (120 Vac 60 Hz) or 7011432-02 (240 Vac 50 Hz), (see Figure A-15). Connect fan ac cable connector P2 to connector J2 on the NIA20 card cage and then join the fan ac cable and the power cord together. Insert the end of the power cord into any available switched outlet on the 861 power controller. Connect the ground wire to the adjacent ground screw on the NIA20 card cage. Use a starwasher to ensure a good electrical connection.

20. Install a Tinnerman nut in hole 13 on the frame and attach the ground cable from the NIA20 card cage to the frame. Use two starwashers to ensure a good electrical connection.

21. Locate the PLI cable, Digital P.N. BC06R-08 (see Figure A-1 for cable route and Figure A-10 for cable connection). Connect one end of the PLI cable (identified by a red line imprinted on top of the cable) to module M3003 and route through the cable strain relief on the NIA20 card cage. The other end of the PLI cable (identified by a red line imprinted on bottom of the cable) to connector J3 on the NIA20 card cage (see Figure A-8). To secure the PLI cable, install adhesive foam, Digital P.N. 1213716-00, within each of the four flat

cable clamps. Install one cable clamp on the side of the card cage, and three across top rear of RH20 card cage door.

22. Reinstall the CPU baffle panel.

23. Locate and connect the BNE3 external cable into connector P1 of the NIA20 current limiter (see Figure A-9) and route down along the side frame and out the bottom of the CPU cabinet.

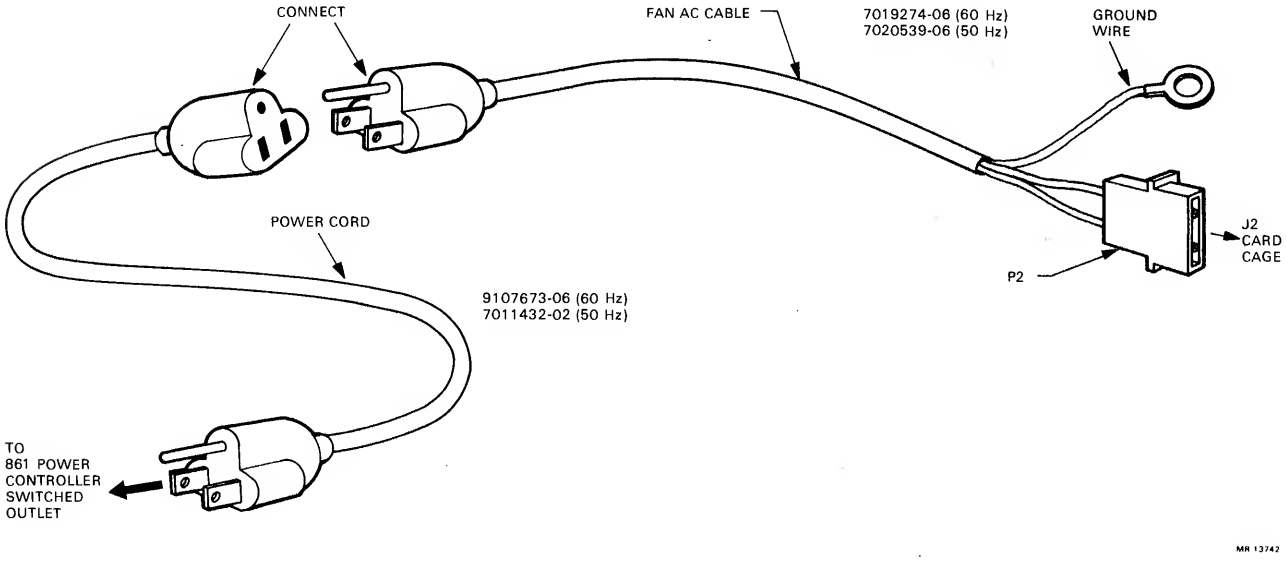


Figure A-15 Fan AC Cable and Power Cord

A.4.6 Installation of KL10 Adapter Board and Blank Module Assembly

The KL10 to NI adapter board, Digital P.N. L0072-00, and the blank module assembly, Digital P.N. 7014103-00, are installed in the NIA20 card cage as follows:

1. The KL10 to NI adapter board and the blank module assembly are installed into the NIA20 card cage by opening its front hinged end panel door.

2. Install the KL10 to NI adapter board (L0072-00) in the right-hand slot.

3. Install the blank module assembly (7014103-00) in the adjacent slot to the left.

A.4.7 Checkout

The physical part of the installation is complete at this point. All that remains is to verify that the system runs properly in the new configuration. Perform the following steps to verify the installation.

1. Verify that the KL10-D is no longer in the override fault state.

2. Power-up the KL10-D.

3. Readjust the 5 V power supply to 5.0 +/- 0.25 V. This adjustment is located on power supply H7420 number 1, H7440 slot 5 (see Figure A-6). This regulator is located nearest the H7420 power supply breaker. The voltage is monitored at the black and red wires on connector J1 of the NIA20 card cage (see Figure A-8).

4. Load and run diagnostic DFPTA for at least five passes in executive mode.

5. Load and run diagnostic DFNIA for at least five passes in executive mode.

6. Enable the operating system.

7. Run diagnostics UETP NIA20 test in user mode for at least four hours.

8. Disable the operating system.

9. Remove all field service packs and tapes from the customer's system and store in a secure area.

10. Transfer/signoff system to customer's authorized representative.

APPENDIX B
INSTALLATION OF NIA20 IN KL10-R

B.1 OVERVIEW

This appendix describes the installation of the NIA20 network interconnect adapter in a KL10-R system. Figures B-1 and B-2 show the NIA20 installed in a KL10-R, rear and front views, respectively. Table B-1 itemizes the NIA20 parts and Table B-2 lists the harness and cable connections used in the NIA20/KL10-R installation.

The NIA20 installation uses assigned slots in RH20 logic assembly positions 4 and 5, with RH20 positions 6 and 7 reserved for installation of a CI20 computer interconnect. A system containing an NIA20 is limited to a maximum of four RH20s. In the installation of an NIA20, a module blank assembly, Digital P.N. 7019266-00, is used to prevent inserting any other module into RH20 position 4 (described in Section A.4.3, instruction 7 -- see Figure A-3).

NOTE

The prior or subsequent installation of a CI20 computer interconnect with an NIA20 requires minor deviations from the following procedures, which are described herein when applicable.

Installation of the NIA20 in an existing system requires implementing the following procedures:

1. Unpacking and checkout of installation kit
2. Preinstallation checkout
3. Backplane wire adds
4. Installation of port modules
5. Installation of power supply regulator
6. Installation of NIA card cage
7. Installation of NIA current limiter
8. Installation of dc power harness
9. Installation of vane switch harness
10. Installation of dc voltage monitor harness and module
11. Installation of PLI bus
12. Installation of fan ac cable and power cord
13. Installation of internal NIA cable
14. Installation of KL10 adapter board and blank module assembly
15. Checkout.

The following sections provide detailed instructions for performing each of these installation procedures.

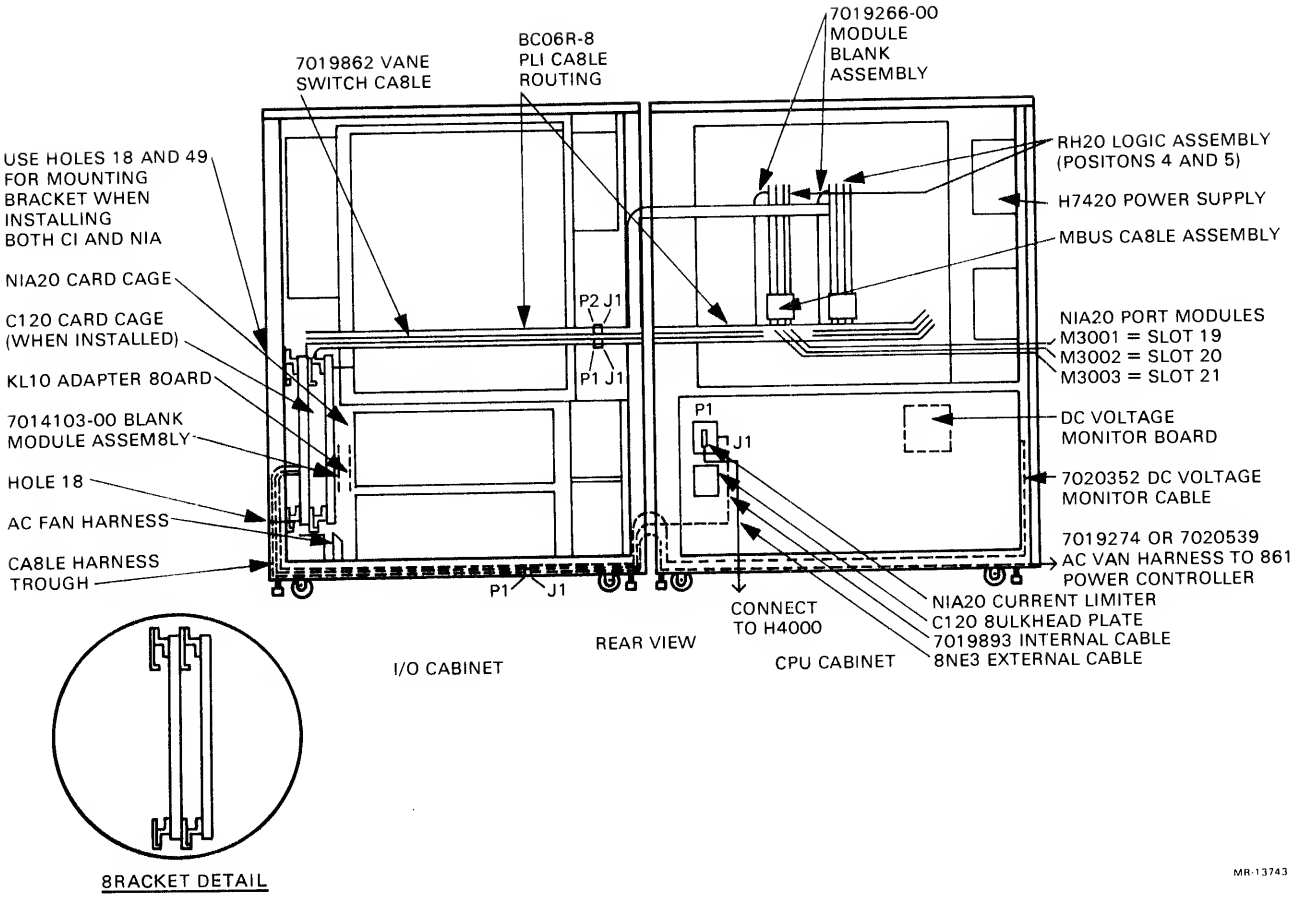


Figure B-1 NIA20 in KL10-R, Rear View

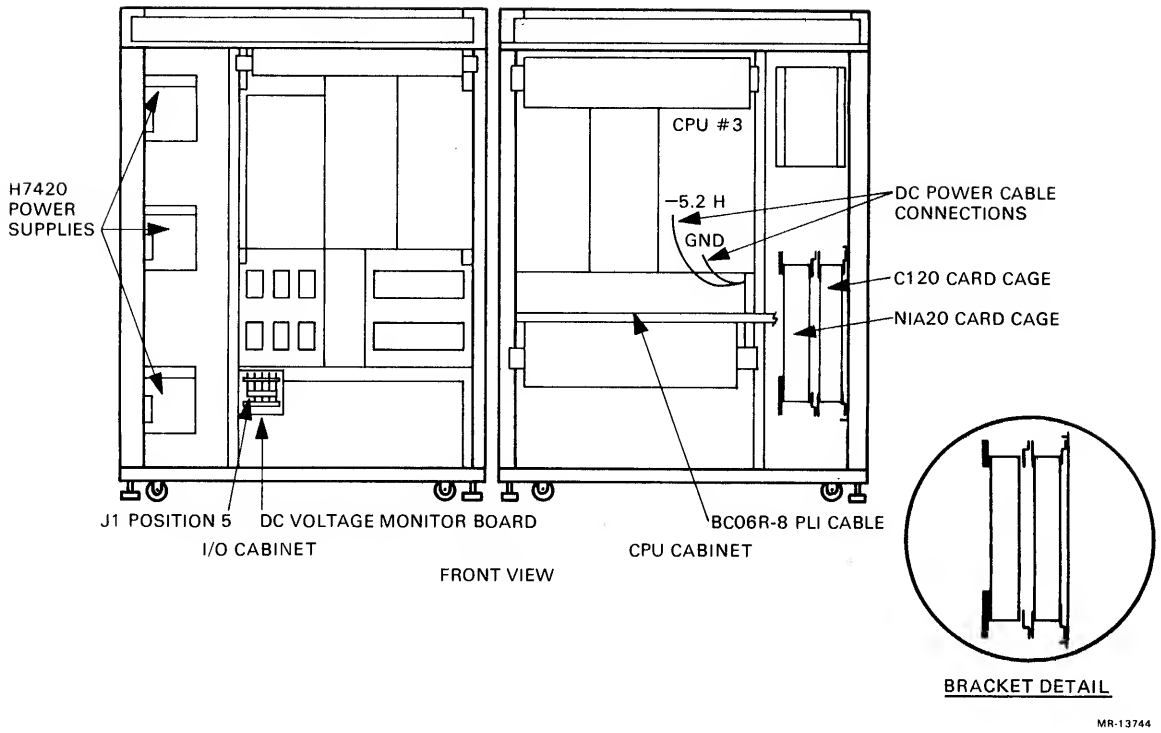


Figure B-2 NIA20 in KL10-R, Front View

Table B-1 NIA20 in KL10-R, Parts List

Line Item	Part No.	Description	Qty
1	7019268-00	Card Cage Assy IPA-20-L	1
2	7019268-01	Card Cage Assy CI20	1
3	-	-	-
4	-	-	-
5	9007786-00	Retainer, U-nut 10-32X	9
6	9006073-01	Screw, mach pan Phil 10-	17
7	9107240-09	Wrap, cable, .250 OD vinyl wht A/R	
8	9105740-55	Wire (wrap) 30 AWG KYNAR UL14 (12 ft)A/R	
9	1213716-00	Spacer, foam Polyu 1/2	5
10	7020539-06	Cable, fan ac	1
11	7019274-06	Cable, fan ac	1
12	7019273-00	Harness DC-5.2 sect N1-2 dc+5	1
13	7019272-00	Harness DC-5.2 sect N1-1 dc+5	1
14	7019893-7L	Cable assy, Ethernet	1
15	BC06R-08	BC06R I/O cable	1
16	7019266-00	Model blank assy	1
17	M3002-00	CI20 microprocessor, multiwire HE	1
18	M3003-00	CI20 CBus/PLI interface, multiwire	1
19	M3001-00	CI20 EBus interface, multiwire H	1
20	9007032-00	Tie, cable bundl. Dia 0-1-3/4"= 101 A/R	
21	1213715-00	Clip, flat cable w/adhesive bk	5
22	H7440-00	POA1 H7440	1
23	L0072	NI20 (KL10 to NI adapter)	1
24	7014103-00	Blank module assy	1
25	9107673-06	Pwr Cord, term 3-14 SJT 115	1
26	7011432-02	Pow cord extension 50 Hz	1
27	9007651-00	Washer, lock external steel	17
28	9006668-00	Washer, flat SST	17
29	7020352-00	Harness, dc voltage monitor	1
30	7019862-00	Harness, vane switch	1
31	5414506-01	Voltage, monitor board	1
32	7019270-1J	Bus, cable, M assy	1
33	3621499-01	Label, DCV monitor CI20	1
34	3613272-00	Label, adhesive back, Mylar cap	1
35	9007031-00	Tie, cable bundl. Dia 0-3/4"=101	36
36	9008264-00	Mount, cable tie, adhesive back	A/R
37	3621498-01	Label, airflow, CPU CI20	1
38	5415695-01	Current limiter	1
39	9006022-01	Screw, mach pan phil 6-	4
40	9006633-00	Washer, lock internal steel	4
41	7020488-00	Cable, short switch vane	1
42	7430278-01	Bracket, mounting	1
43	9006659-00	Washer, flat S/PAS	2
44	7021448-5C	Cable dc voltage monitor Sect.1	1
45	3617674-00	Label, serial/power w/o UL + CSA	1

Table B-1 NIA20 in KL10-R, Parts List (Cont)

Line Item	Part No.	Description	Qty
46	3617674-01	Label, serial/power w/UL & CSA	1
47	3617880-09	Label, class A subassembly	1
48	3621501-02	Label, module location, NI20	1

Table B-2 NIA20 in KL10-R, Harness and Cable Connections

Parts List Item No.	Harness Terminals Point	Connection	Connection	Remarks
		Harness Connections		
12	_	P1	NIA20 BP J2	
	=	P3	NIA20 BP J1	
	5	-	CPU #3 BF GND	
	6	-	CPU #3 BP -5.2	
	-	P2	Sect N-2 J1	Parts list item 13
13	-	P1	H7440 J1	See Figure B-5
	7	-	See Figure B-10	
	8	-	See Figure B-10	
30	-	P1	Fan Brkt J1	See Note 1
	-	P2	See Figure B-10	See Note 1
	-	J1		See Notes 1 and 2
	-	P3	NIA20 BP J6	See Notes 1 and 3
29	-	P2	NIA20 BP J5	
	-	P1	J1	Parts List Items 31 and 44
	6	-	+5V Mon Bd J1-5	See Figure B-2
41	P3	P1	CI20 cable vane Switch	See Note 4
	P2	-	NIA20 fan brkt J1	See Note 4
	P1	-	CI20 fan brkt J1	See Note 4
44	-	J1	P1	See Figure B-13
	-	P1	Mon Bd J1	Parts List Item 31

NOTES:

1. Items not needed when CI kit is installed:

Item	Qty	Part No.	Description
5	6	9007786-00	Retainer, U-nut
6	6	9006073-01	Screw, mach pan phil 10-

Table B-2 NIA20 in KL10-R, Harness and Cable Connections (Cont)

27	6	9007651-00	Washer, lock external steel
28	6	9006668-00	Washer, flat SST
30	1	7019862-00	Harness, vane switch

2. J1 on parts list Item 30 (harness, vane switch) connects with existing connector P4 (see Figure B-12).

3. Relocate cable from CI20 card cage at J6 connector and insert into NIA20 card cage J6 connector.

4. Parts list Item 42 used when CI20 and NIA20 are installed together.

Cable Connections

Parts List Item No.	From Unit	Location	Ref. Desig.	To Unit	Location	Ref.Desig.	Remarks
10	Cd.Cage	Gnd	Gnd	25 or 26	J1	P1	NIA20-CB
	Fan Brkt	J2	P2				
14	Cd.Cage	J4	P1	38		P2	
25 or 26	Item 10 or 11	P1	J1	861 PC		P1	Connect any available switched outlet
1/2	Cd.Cage	Gnd	-	Cabrail	Hole 1	-	
15	CI20 B	J3	P1 stripe	RH20 DTE	M3003 J2	P2 stripe	
			down	CC		up	
32	M3003	J1	P3	M3002	J1	P2	
	M3001	J1	P1				
8	RH20 BP	B1 N1	-	RH20 BP	B13B1	-	
		B13B1	-		B19B1	-	
		C10L2	-		B13F2	-	
		B13B2	-		B19B2	-	
		C10K2	-		B13U1	-	
		B13U1	-		B19U1	-	
		C10T2	-		C13B1	-	
		C13B1	-		C19B1	-	
		C12H2	-		C13N1	-	
		C13N1	-		C19N1	-	
		C12L1	-		C13F2	-	
		C13B2	-		C19B2	-	
		C14H2	-		A15R2	-	

Table B-2 NIA20 in KL10-R, Harness and Cable Connections (Cont)

		C14F1	-		F15A1	-	
		A14J2	-		A15E1	-	
		C14P1	-		A15D2	-	
		C14K2	-		A15S2	-	
		B14J1	-		B15A1	-	
		C20H2	-		A21R2	-	
		C20F1	-		F21A1	-	
		A20J2	-		A21E1	-	
		C20P1	-		A21D2	-	
		C20K2	-		A21S2	-	
		B20J1	-		B21A1	-	
11	Cd.Cage	Gnd	Gnd	25 or 26	J1	P1	NIA20-CA
	Fan Brkt	J2	P1				

B.2 UNPACKING AND CHECKOUT

Before unpacking any equipment, move all boxes into the computer area. Check the shipment against the packing list to be sure that all boxes were sent. If any boxes are missing, contact the customer and the branch field service manager. Check that all boxes are sealed, and there is no sign of external damage, such as dents, holes, or damaged corners.

If any boxes are open or damaged, document it on the installation or field service report and inform the customer. Open the boxes one at a time, starting with the box marked "READ ME FIRST" and find the packing slip. Check the contents of the box against the packing slip and examine each item for damage. Note missing or damaged items on the installation report or field service report.

This completes the unpacking and checkout phase. Advise the branch field service manager of any problems during this phase. If any items are damaged, the branch field service manager may want the customer to file an insurance claim. For missing items, the branch field service manager should get a short-ship request.

B.3 EQUIPMENT NEEDED FOR INSTALLATION AND CHECKOUT

The following equipment is required for installation and checkout of the NIA20:

1. Wire wrap tool No. 30 AWG, Digital P.N. 29-18301
2. Wire unwrapping tool, No. 30 AWG, Digital P.N. 29-13513
3. Regular Phillips screwdriver
4. Tektronix 475 oscilloscope or equivalent (100 MHz)
5. KLAD pack
6. Scope, digital voltmeter.

B.4 INSTALLATION PROCEDURE

B.4.1 Preinstallation Checkout

Before performing the installation, verify that the currently configured system is operating properly to preclude the possibility of present system problems being ascribed to the NIA20 after its installation.

1. Remove all customer media, to minimize the possibility of corrupting customer data.

2. Mount the supplied KLAD pack, bring up the diagnostic monitor, and run the B string to verify that the system is working properly.

3. Power-down the system.

4. Verify that the system has a M8532-YA board installed. If not, replace the currently installed M8532 with a M8532-YA.

5. RH20 positions 4 and 5 are used for the NIA20. If there is an RH20 in position 4, remove it. If there is an RH20 in position 5, leave it temporarily installed and perform diagnostic DFRHB to verify the reliability of the backplane wiring. If there is no RH20 in position 5, relocate a module from one of the other RH20 positions to position 5.

6. Power-up the system and run diagnostic DFRHB. This verifies that the backplane wiring of RH20 position 5 is functional. Power-down the system and reinstall the RH20 in its original position.

7. Perform diagnostic DFRHB also in RH20 position 7 to verify the reliability of existing backplane wiring in RH20 positions 6 and 7 for CI20 installation, before backplane wiring modifications for the NIA20 are implemented.

8. Power-down the system and reinstall the RH20 in its original position.

B.4.2 Backplane Wire Adds

For the NIA20 installation, 24 new wires must be added to RH20 backplane positions 4 and 5. An examination of the RH20 backplane must be performed to confirm the physical addition of the wire wraps listed in Table B-3. The table contains a Check column for the wire installer to record installation progress.

#59
64..

To prepare the wire adds, strip approximately 1 inch of insulation from the wire to allow sufficient turns to be made on the wire wrap post. After each wire is added, enter a check in the blank adjacent to the wire listing in Table B-3.

To assure the reliability of the new wiring, an ohmmeter check of each new wire add should be performed by a person other than the wire installer.

Table B-3 NIA20 in KL10-R Wire Adds

Signal Name	From	To/From	To	Check
EBUS D11 L	B10N1	B13B1	B19B1	
EBUS D12 L	C10L2	B13B2	B19B2	
EBUS D13 L	C10K2	B13U1	B19U1	
EBUS PARITY L	C10T2	C13B1	C19B1	
EBUS PI00 L	C12H2	C13N1	C19N1	
EBUS PARITY ACTIVE L	C12L1	C13B2	C19B2	
MPR7 MWBUSCTLFLD01 H	C14H2	A15R2		
MPR7 MWMGCFLD08 H	C14F1	F15A1		
MPR7 MWTIMEFLD H	A14J2	A15E1		
CBI1 CLK2 L	C14P1	A15D2		
CBI2 CLK4 L	C14K2	A15S2		
CBI2 CCCHANERR L	B14J1	B15A1		
MPR7 MWBUSCTLFLD01 H	C20H2	A21R2		
MPR7 MWMGCFLD08 H	C20F1	F21A1		
MPR7 MWTIMEFLD H	A20J2	A21E1		
CBI1 CLK2 L	C20P1	A21D2		
CBI2 CLK4 L	C20K2	A21S2		
CBI2 CCCHANERR L	B20J1	B21A1		

B.4.3 Installation of Port Modules

Protective backing is placed on the lower third noncomponent side of each port module and the upper third of the noncomponent side of the M3002. As the modules are inserted and/or removed, the protective backing protects the MBus and PLI bus cables. The protective backing should not interfere with the card guide or cover the gold finger contacts on the module. Insert the port modules as follows:

1. Connect MBus cable, Digital P.N. 7019270-1J. Be sure to orient the cable so that the flat wire comes out of the cable header away from the board, as shown in Figure B-3.

2. Insert the M3001 EBus interface/port ALU module in the rightmost slot of RH20 position number 5 (slot 19, looking at the backplane from the module side). The arrow on cable should be aligned with the arrow on the board connector.

3. Connect the MBus cable to the M3002 port microprocessor module as shown in Figure B-3.

4. Insert module M3002 in slot 20 to the left of the installed M3001 as shown in Figure B-3.

5. Connect the MBus cable to the M3003 module as shown in Figure B-3.

6. Install the M3003 CBus/PLI interface module in slot 21, which is located to the left of the installed M3002.

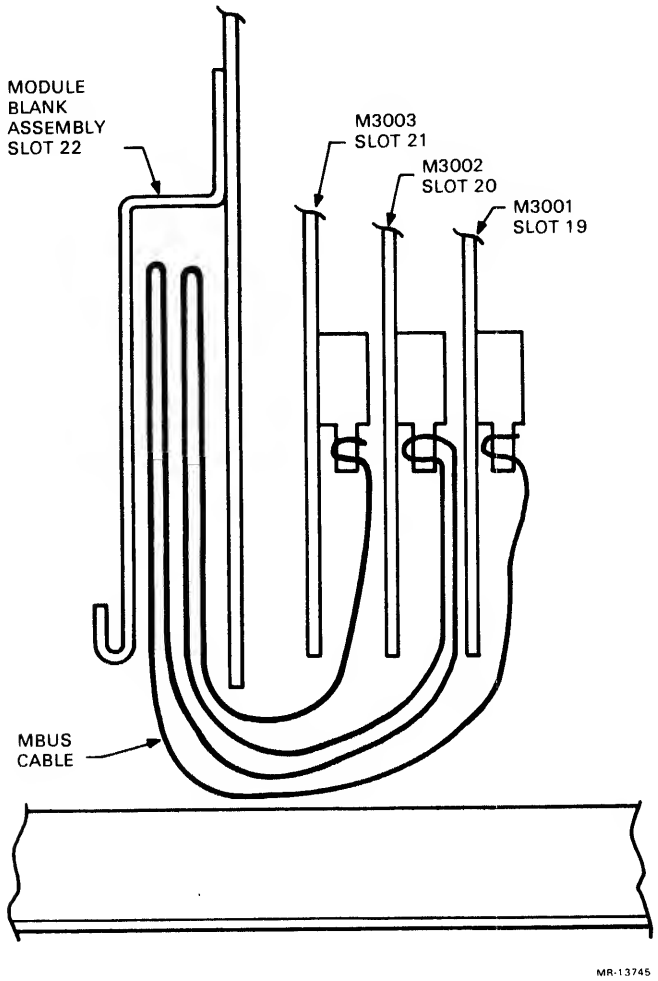


Figure B-3 MBus Cable Interboard Connection, Top View

7. Install the module blank assembly, Digital P.N. 7019266-00, in RH20 position 4, slot 22. This assembly blocks slots 22, 23, and 24. It prevents modules from being plugged into RH20 position 4 and provides a baffle for system cabinet airflow.

8. Perform an ohmmeter check between PT17U and ground to verify that there are no shorts to ground.

9. Fold the MBus cable into the module blank assembly as shown in Figure B-3.

10. Close the module door.

11. Attach the self-sticking module utilization decal, Digital P.N. 3622344-02, on the upper rear baffle panel.

12. Power-up the KL10.

13. Readjust the existing +5 V power supply to 5.0 $\pm$ 0.25 V. This adjustment is located on H7420 number 1 in H744 number 4. The location of this regulator is farthest away from the circuit breaker. The voltage is monitored at +5E, between PT17U and ground.

14. Type MR (CR) with KLDCP loaded and running; then type FX1 (CR) in response to the command prompt, as shown below:

```
>. MR (CR)

>. FX1 (CR)
```

15. De-skew the port modules using a Tektronix 475 or equivalent 100 MHz minimum oscilloscope by performing the following steps (see Figures B-4 and B-5):

16. Connect channel 1 of the oscilloscope to MTR MBOX CLK H, 4D33P1, on the CPU backplane. Use a ground clip.

17. Set the time base to 20 ns.

18. Set channel 1 vertical gain to 0.5 V/division. Set the ground reference to 1.3 V above the horizontal center level of the oscilloscope (MTR MBOX CLK H is an ECL signal).

19. Set the oscilloscope sync to positive external.

20. Connect external sync input to CHT0 H, 4B09K1 on the CPU backplane. Use a ground clip.

21. Connect channel 2 to CDS1, EBUS CLK L, 2A21F1 on the I/O backplane. Set the channel 2 vertical gain to 0.5

V/division. Use a ground clip. To measure TTL voltages, set the ground reference to 1.5 V below the horizontal center line of the oscilloscope.

22. Press the trigger view switch of the oscilloscope and display the external sync. Adjust the display, so that the rising edge of the external sync aligns with the vertical center line of the oscilloscope.

23. Display MBOX CLK H, channel 1. Identify the rising edge of MBOX CLK H that occurs prior to the vertical center line of the oscilloscope. Display channel 1 and channel 2.

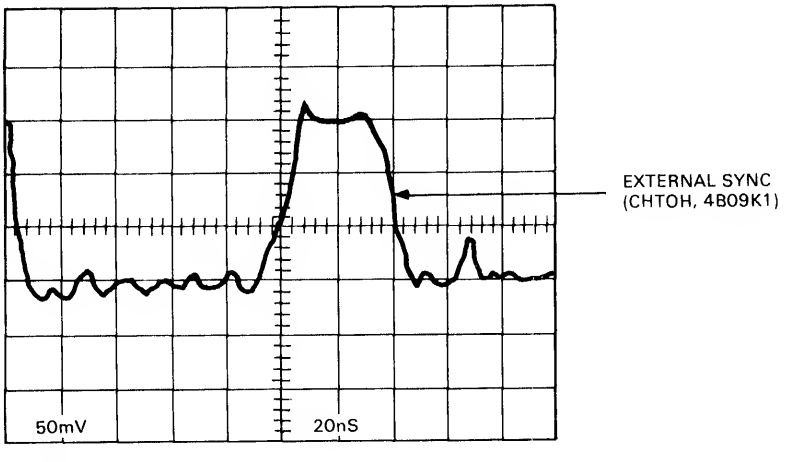
24. Put the KL10-R in the override fault state. Remove the I/O rear door to access the I/O backplane.

25. Locate the third the bottom potentiometer on the clock module (M8559) in slot 12 of the I/O backplane. Using this potentiometer, adjust the falling edge of channel 2, EBUS CLK L so that it crosses the rising edge of MBOX CLK H. This crossing occurs on the horizontal center line of the oscilloscope.

26. Disconnect all probes.

27. Mount the KLAD pack on the front end RP06.

28. Load and run diagnostic DFPTA to verify proper functioning of the modules. If the modules fail, troubleshoot as directed by the diagnostic. If the modules are functioning properly, continue with the installation.



EXTERNAL SYNC (CHTO H)

Figure B-4 NIA20 De-skew Timing. External Sync (CHT0 H)

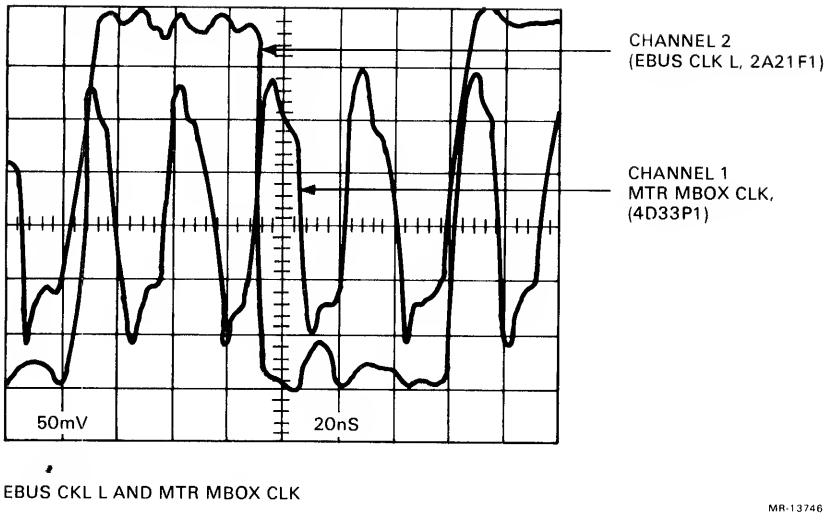


Figure B-5 NIA20 De-skew Timing. EBUS CLK L and MTR MBOX CLK

B.4.4 Power Supply Regulator Installation

Three H7420 power supplies are located on the I/O cabinet side wall as shown in Figure B-6. The H7440 regulator to be added is installed in the upper H7420 power supply location. This additional +5 V regulator is required to support the NIA20 card cage and is installed as follows (see Figure 3-6):

1. Remove the spare slot filler panel from slot 5 of the H7420 number 1 power supply. Save all existing hardware.

2. Take the new H7440 regulator from the kit and install the H7440 in slot 5 of H7420 number 1, using two screws on top and one thumbscrew at the bottom. Some systems may use H744 or H7440 regulators.

B.4.5 Installation of NIA20 Card Cage/Internal Cable

NOTE
When a CI20 is installed, the NIA20 is mounted as shown in Figure B-1.

To install the NIA20 card cage shown in Figure B-7 and the internal NIA20 cable, perform the following operations:

1. Install the two NIA20 mounting brackets, DEC part number 7430278-01, shown in Figure B-8 as follows:

   Remove and reposition any tie-wrapped cables from the left frame member of the CPU cabinet as viewed from rear (see Figure B-1) to accommodate the NIA20 mounting brackets and card cage.

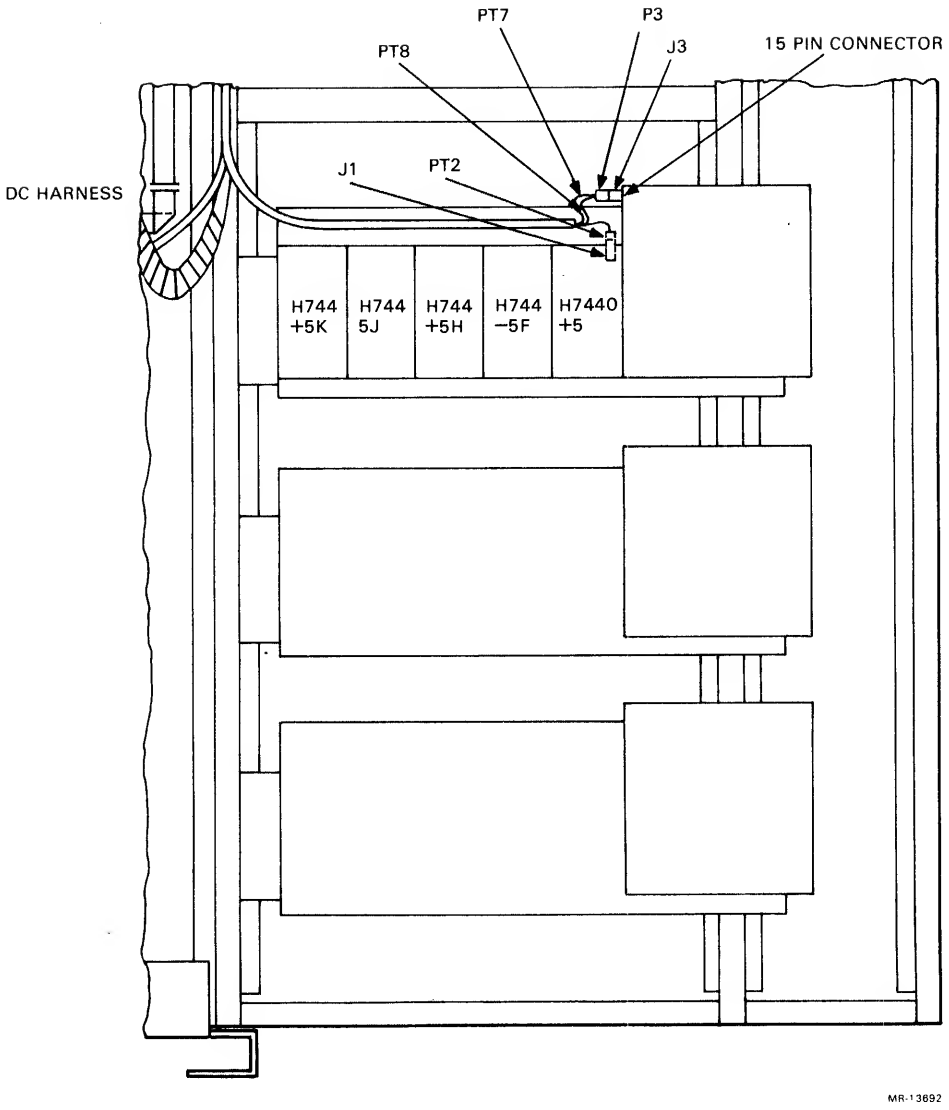


Figure B-6 H7420 Power Supply

Install 4 U-nuts (Tinnerman nuts), Digital P.N. 9007786-00, on the left side frame member of the CPU cabinet (viewed from the rear). Insert the Tinnerman nuts into frame holes 18 and 49 on each vertical side frame member counting up from the cabinet bottom. Two Tinnerman nuts are inserted into each side frame member.

Check that the lip of each NIA20 mounting bracket, Digital P.N. 7430278-01 (see Figure B-8), is at the bottom (used only in combined NIA20/CI20 installations). Use a total of four 10/32 one-half inch Phillips panhead machine screws, Digital P.N. 9006073-01 and four No. 10 star lockwashers, Digital P.N. 9007651-00.

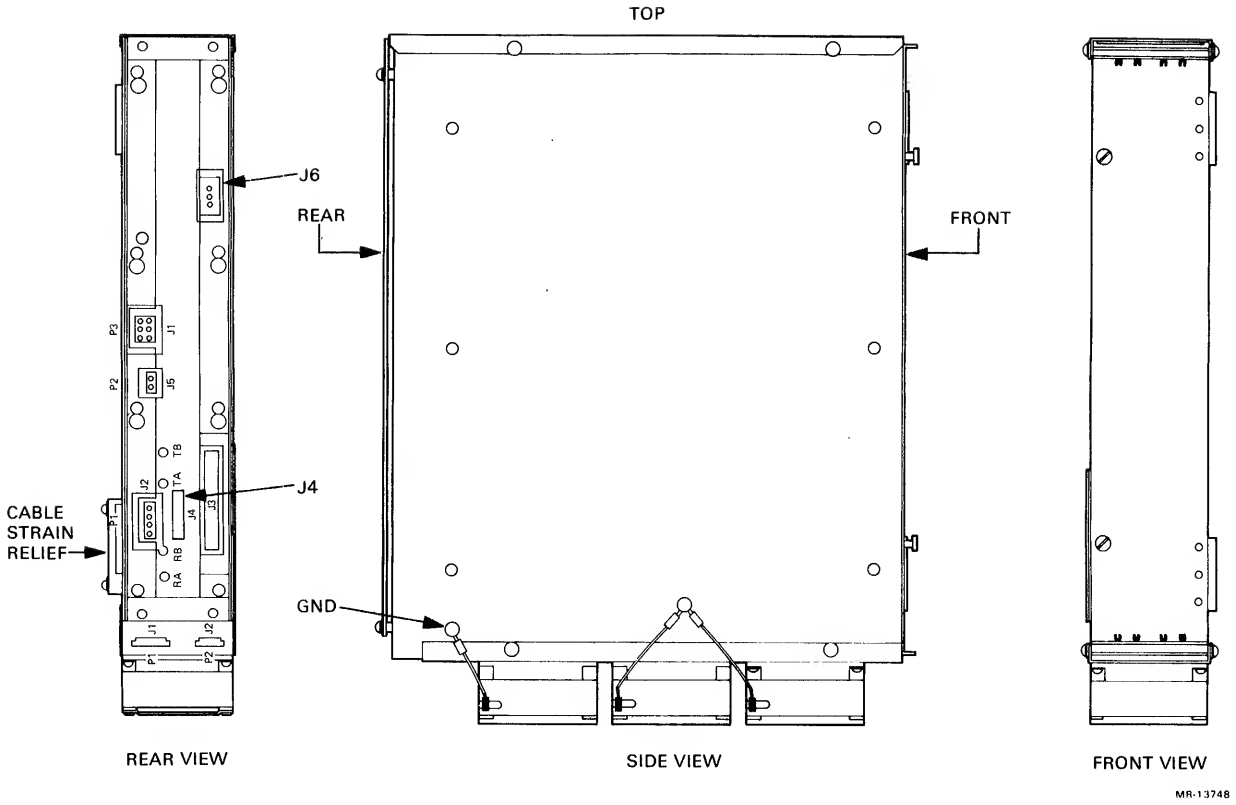


Figure B-7 NIA20 Card Cage Views

2. Locate the NIA20 internal cable strain relief (white) on the left side of NIA20 card cage as viewed from the rear (see Figure B-7). Because of the inaccessible location of this strain relief once the card cage is mounted, the internal cable is be routed through it before the card cage is installed.

3. Locate the eight-foot internal NIA20 cable, Digital P.N. 7019893-7L.

4. Prepare the internal NIA20 cable for installation by positioning a stick mount on the lower left frame member of the I/O cabinet (see Figure B-9) located above the current limiter and reserved CI20 plate location.

5. Route the eight-foot internal NIA20 cable through the white plastic strain relief on the NIA20 card cage. Allow enough slack to connect the internal cable to the NIA20 card cage backplane and tighten the strain relief.

6. Connect the internal NIA20 cable to the J4 connector (see Figure B-7) on the rear of the NIA20 card cage and route the cable as shown in Figure B-1. The cable connector engages a detent when properly seated.

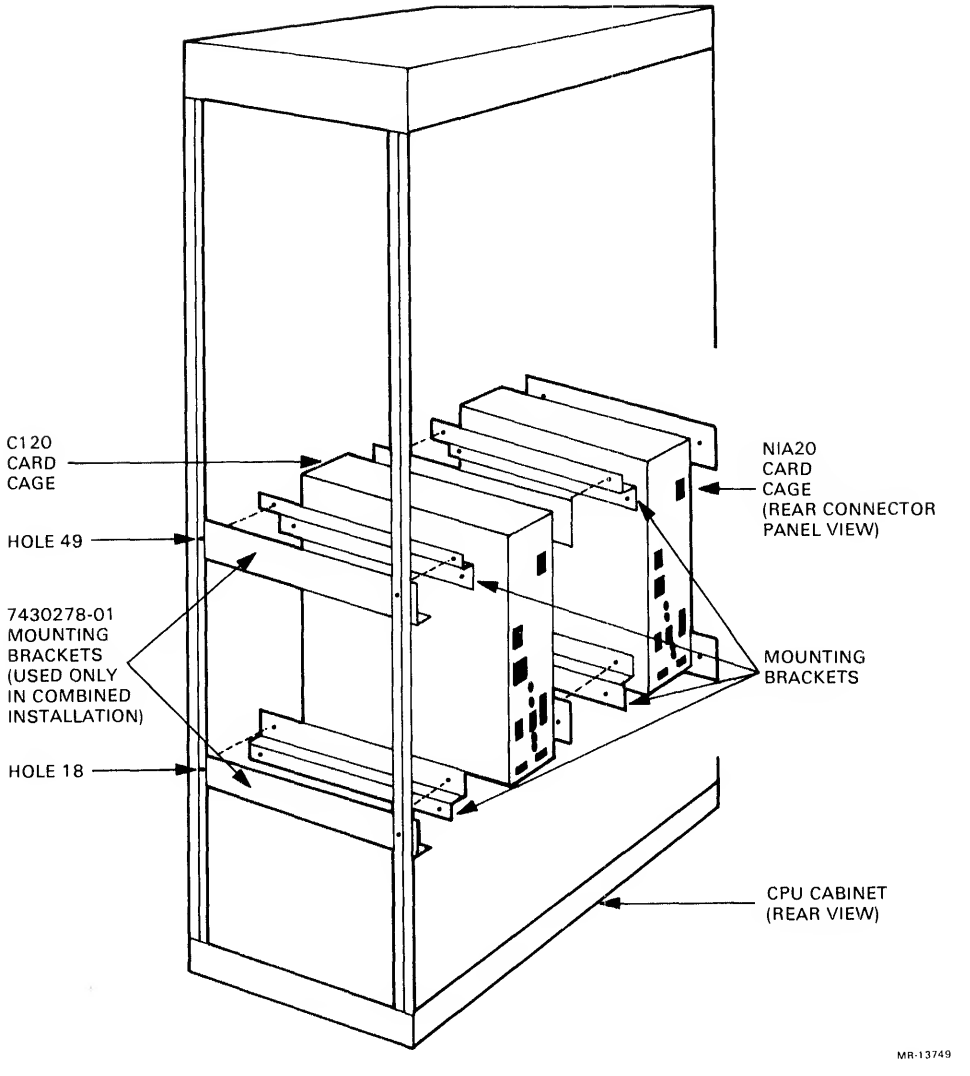


Figure B-8 NIA20 Card Cage in KL10-R

7. Mount the NIA20 card cage on the two NIA20 mounting brackets (see Figure B-8), using four 10/32 screws, external lockwashers, and flat washers. Hang the NIA20 card cage on the top two screws; then install the bottom two screws.

8. Install the NIA20 card cage ground cable as follows:

   Install a Tinnerman nut in hole 1 of the left side frame member of the CPU cabinet (viewed from the rear).

   Connect the ground cable on the left side frame member by inserting a screw and using a starwasher on each side of the ground cable.

   Attach a ground label, 3613272-00, closest to hole 1.

9. Run the internal NIA20 cable to the previously positioned stick mount and insert the other end of the cable into the rear J1 connector of the NIA20 current limiter.

B.4.5.1 **Installation of NIA20 Current Limiter** -- The NIA20 filler plate, Digital P.N. 7429334-01, mounted above the CI20 plate located on the bottom panel of the I/O cabinet is replaced by the NIA20 current limiter, Digital P.N. 5415695-01 (see Figure B-9). Install the NIA20 internal cable and current limiter as follows:

1. Locate the NIA20 filler plate from the rear of the I/O cabinet.

2. Remove the four screws holding the filler plate and save screws to install the NIA20 current limiter using its aligned threaded holes.

3. Connect the internal cable to the rear J1 connector and also connect the BNE3 external (Ethernet transceiver) cable to the front P1 connector located on the NIA20 current limiter (see Figure B-9).

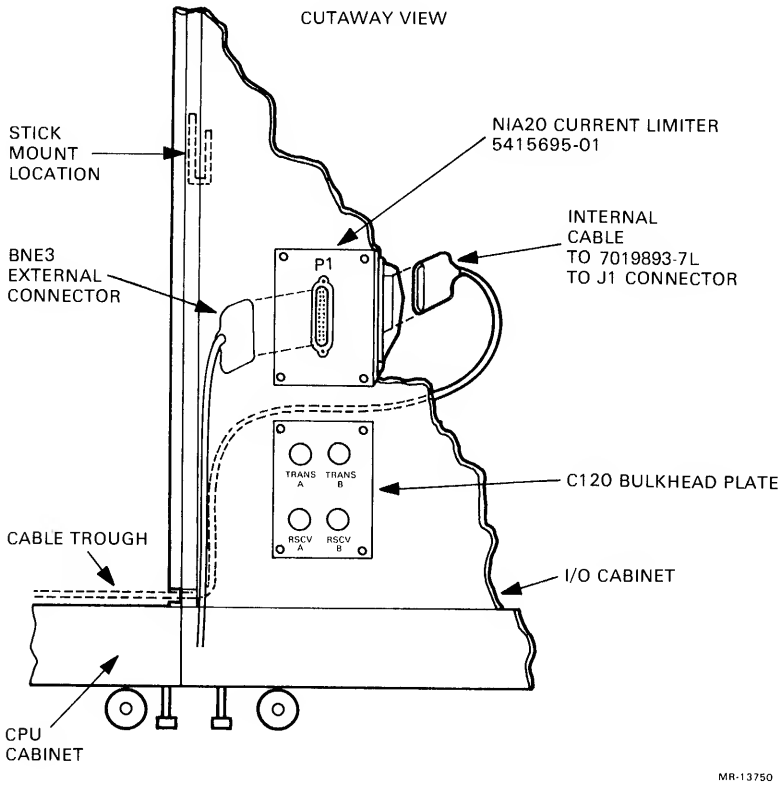


Figure B-9 NIA20 Current Limiter, Cutaway View

B.4.5.2 **Harness Installation** -- The following harnesses are to be installed:

1. DC power harness (two sections)
2. Vane switch cable
3. DC voltage monitor cables (two sections)
4. Fan ac cable and power cord
5. PLI bus
6. BNE3 external cable.

Figure B-10 shows a diagram of the harness and cable interconnections. The harnesses are installed as follows:

1. Install tie-wraps approximately eight inches apart on all harnesses. When routing cables close to internal assemblies, use spiral wire-wrap to protect the cables.

2. Locate the two sections of the dc power harness, Digital P.N. 7019272-00, and 7019273-00 (see Figure B-11).

3. Attach one-half of dc power cable, Digital P.N. 7019272-00, connecting P1 of dc power cable into connector J1 of the NIA20 card cage backplane (see Figure B-7). Next, connect P3 of of dc power cable into J2 of the NIA20 card cage. Locate the black and blue wires labeled PT5 and PT6 of the dc power cable and connect the black wire to -5.2 ground; then connect the blue wire to -5.2H in the CPU cabinet (see Figure B-2).

4. Locate the other half of the dc power cable, Digital P.N. 70-19273-00 (see Figure B-11), and join its end labeled J1 to P2 on the other half of the cable (Digital P.N. 7019272-00).

5. Tie-wrap the new harness to the existing KL10-R CPU power harness and route through the cabinet floor as shown in Figure B-1. Use spiral wrap along the harness where it contacts the side of the I/O frame member nearest the H7420 power supplies (see Figure B-1).

6. Locate the red and white wires labeled PT7 and PT8 at the other end of the harness (see Figure B-11). Disconnect P3 from power supply H7420 number 1, then connect PT7 and PT8 to pins 3 and 4, respectively, on P3 of the H7420. Then reconnect P3 to the H7420.

7. Connect P2 of the dc power cable, Digital P.N. 7019273-00 (see Figure B-11), to connector J1 of the H7440 regulator (see Figure B-6).

8. Locate the vane switch cable, Digital P.N. 70-19862-01 (see Figure B-12). Connect P1 of the vane switch cable to connector J1 on the card cage (see Figure B-7). Use stick

mounts and spiral wire-wrap as needed to route and protect the vane switch cable.

NOTE

When a CI20 is installed, the shorter of the two supplied vane switch cables is used in a combined CI20/NIA20 installation. Consult the CI20 reference manual (Digital order number EK-CI20-RM-001) for other applicable CI20 installation procedures.

9. Connect P3 of the vane switch cable to connector J6 on the card cage.

10. Remove original KL10 CPU vane switch cable (P4) and connect this to the NIA20 vane switch cable J1 (see Figure B-13).

11. Connect P2 of the vane switch cable to the original KL10 CPU vane switch assembly.

12. Apply the CPU/NIA20 air flow fault decal over the existing CPU air fault message decal on the 863 fault switch.

13. Locate the two sections of the dc voltage monitor cable, Digital P.N. 7020352-00 and 7021448-5C (see Figure B-14).

14. Connect P2 of the dc voltage monitor cable (Digital P.N. 7020352-00) to connector J5 on the card cage backplane and route the other end (P1) along the inside cabinet top cable trough (see Figure B-1) connecting with J1 of the second section of the dc voltage monitor cable (Digital P.N. 7021448-5C) which has its P1 connected to the J1 on the new dc voltage monitor board (Digital P.N. 5414506-01).

15. Locate the switches on the voltage monitor board. Only switch 1 should be on, while all other switches should be off. Insert the dc voltage monitor board into the +5 V slot (see Figure B-2) of the dc voltage monitor card cage.

16. Attach the monitor panel decal, Digital P.N. 3621499-01, indicating the RH20 slot position used for the NIA20 voltage monitor board.

17. Connect the remaining single orange wire of the P1 end of the dc voltage monitor cable (Digital P.N. 70221448-5C) to a location adjacent to the existing orange wire on the dc voltage monitor board zone +5L.

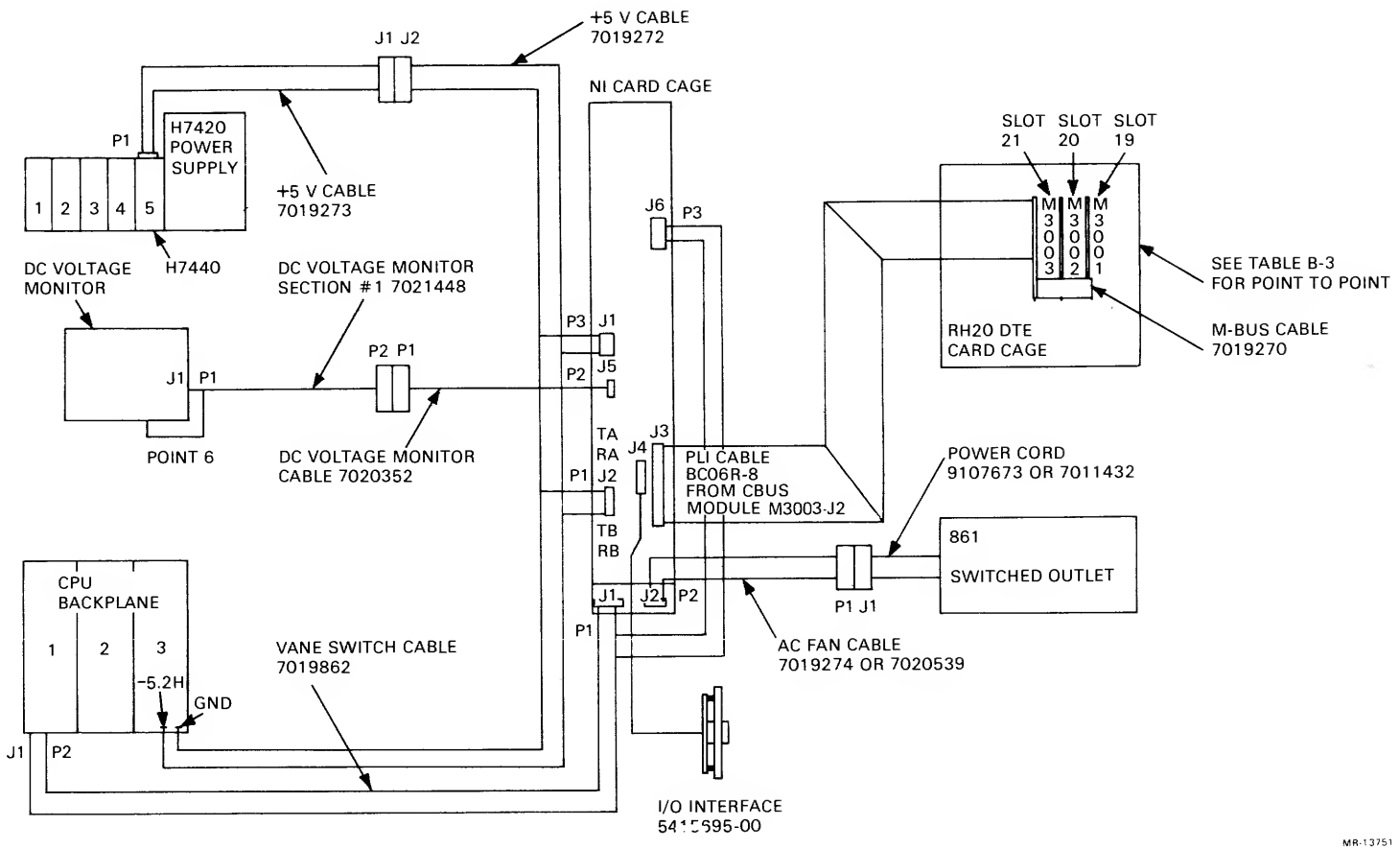


Figure B-10 NIA20 Harness and Cable Interconnection Diagram

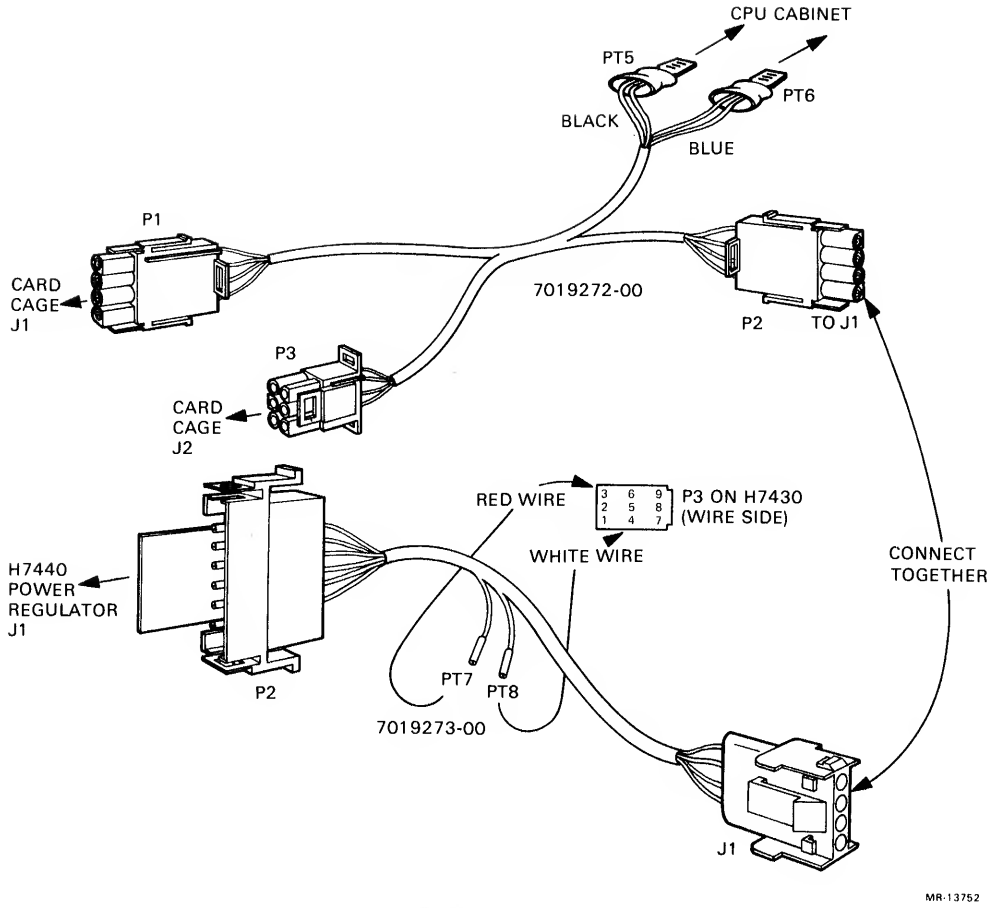


Figure B-11 DC Power Cable

18. Tie-wrap the dc voltage monitor and vane switch harnesses to the dc power cable. Use adhesive-backed square cable mounts to support the harness on the side of the NIA20 card cage.

19. Locate the fan ac cable, Digital P.N. 7019274-06 (120 Vac 60 Hz) or 7020539-06 (240 Vac 50 Hz), and the power cord, Digital P.N. 9107673-06 (120 Vac 60 Hz) or 7011432-02 (240 Vac 50 Hz) shown in Figure B-15. Connect P2 of the fan ac cable to connector J2 on the card cage and then join the fan ac cable to the power cord. Insert the other end of the power cord to any available switched outlet on the 861 power controller. Connect the ground wire to the adjacent side ground screw on the NIA20 card cage. Use a starwasher to ensure a good electrical connection.

20. Install a Tinnerman nut in hole 13 on the frame and attach the ground cable from the NIA20 card cage to the frame. Use two starwashers to ensure a good electrical connection.

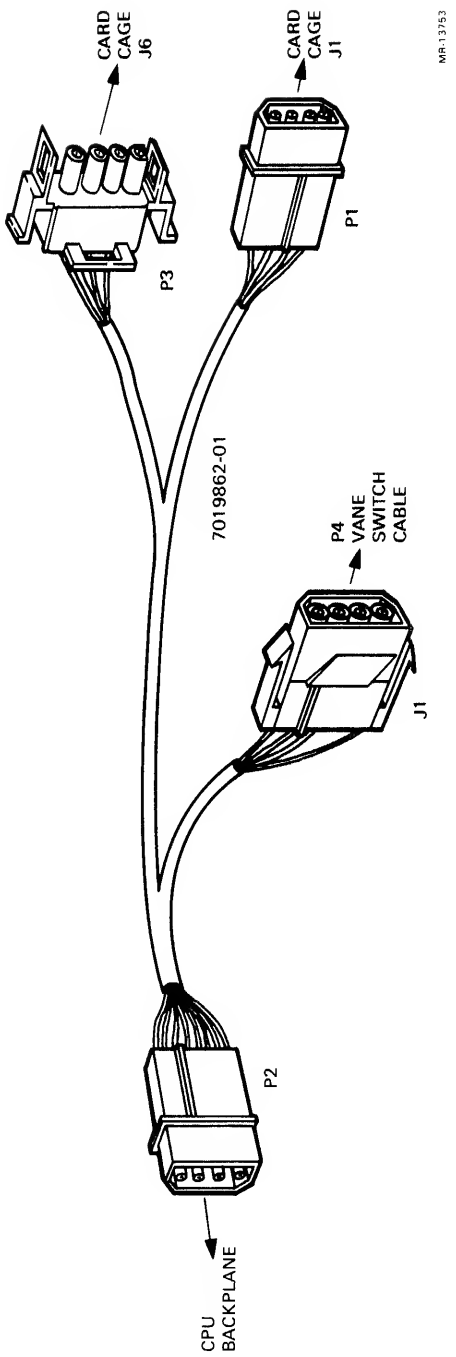


Figure B-12 Vane Switch Cable

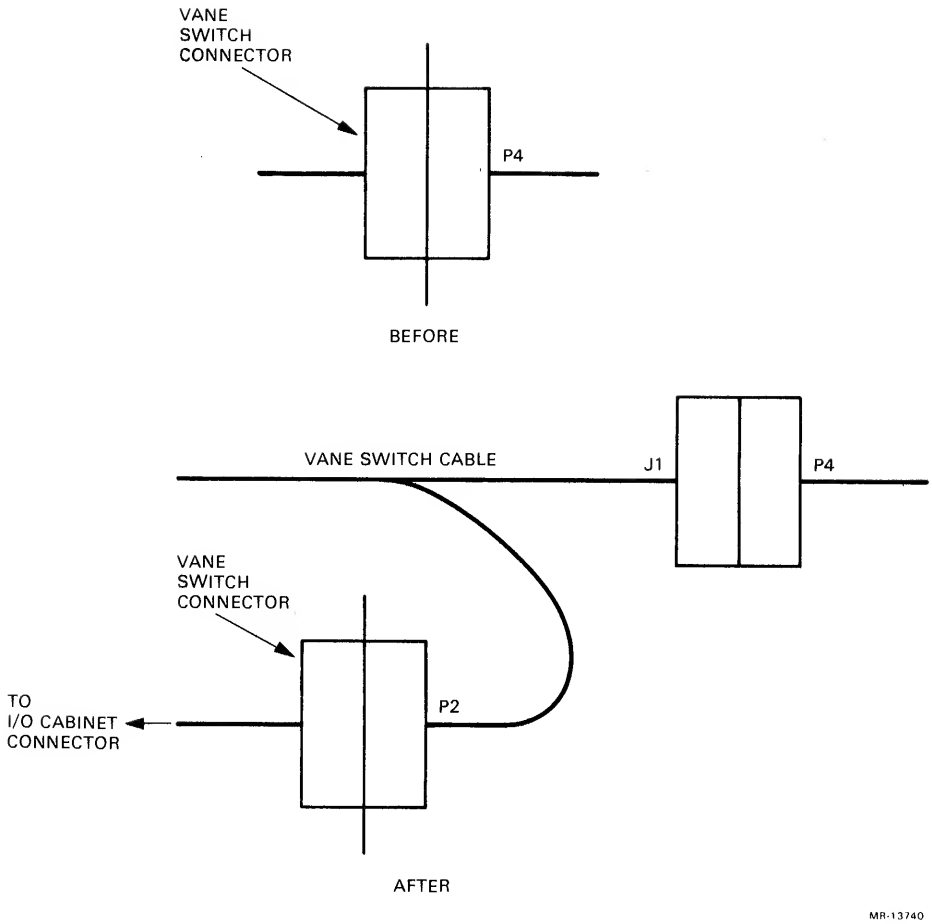


Figure B-13 Vane Switch Harness Installation

21. Locate the PLI cable, Digital P.N. BC06R-08 (see Figure B-1 for cable route and Figure B-10 for cable interconnection). Connect one end of the PLI cable (identified by a red line imprinted on top of the cable) to module M3003, and route through cable strain relief on the NIA20 card cage. The other end of the PLI cable (identified by a red line imprinted on the bottom of the cable) to connector J3 on the NIA20 card cage (see Figure B-7). To secure the PLI cable, install adhesive foam, Digital P.N. 1213716-00, within each of the four flat cable clamps. Install one cable clamp on the side of the CPU card cage, and three cable clamps across the top rear of air shroud assembly.

22. Replace the I/O module door.

23. Locate the BNE3 external cable. Route the BNE3 external cable along the bottom of the CPU and I/O cabinets and then out the bottom of the I/O cabinet. Connect the cable to the front P1 connector on the NIA20 current limiter (see Figures B-1 and B-9).

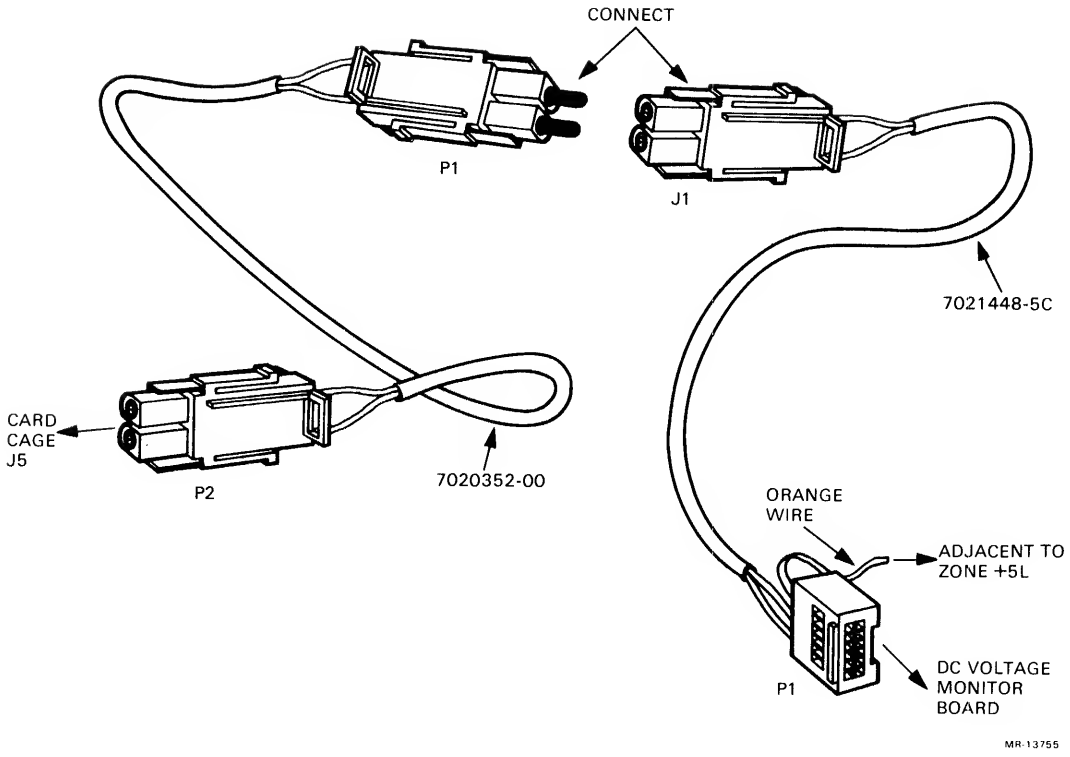


Figure B-14 DC Voltage Monitor Cable

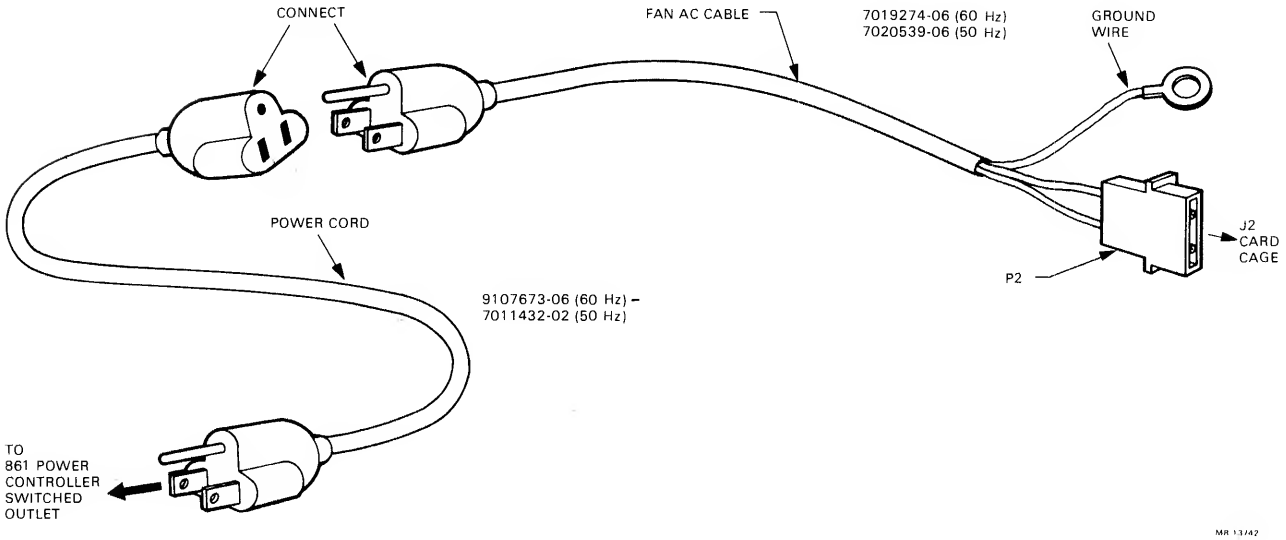


Figure B-15 Fan AC Cable and Power Cord

B.4.6 Installation of KL10 Adapter Board and Blank Module Assembly

The KL10 to NI adapter board, Digital P.N. L0072-00, and the blank module assembly, Digital P.N. 7014103-00, are installed in the NIA20 card cage as follows:

1. The KL10 to NI adapter board and the blank module assembly are installed into the NIA20 card cage from the rear of the CPU cabinet.

2. Install the KL10 to NI adapter board (L0072-00) in the ~~right~~ left-hand slot.

3. Install the blank module assembly (7014103-00) in the adjacent slot to the ~~left.~~ RIGHT.

B.4.7 Checkout

The physical part of the installation is complete at this point. All that remains is to verify that the system runs properly in the new configuration. Perform the following steps to verify the installation.

1. Verify that the KL10-R is no longer in the override fault state.

2. Power-up the KL10-R.

3. Readjust the 5 V power supply to 5.0 $\pm$ 0.25 V. This adjustment is located on power supply H7420 number 1, regulator H7440 slot 5 (see Figure B-6). This regulator is located nearest the H7420 power supply breaker. The voltage is monitored at the black and red wires on connector J1 of the NIA20 card cage (see Figure B-7).

4. Load and run diagnostic DFPTA for at least five passes in executive mode.

5. Load and run diagnostic DFNIE for at least five passes in executive mode.

6. Load and run diagnostic DFNIA for at least five passes in executive mode.

7. Enable the operating system.

8. Run diagnostics DFPTA in user mode for at least five passes.

9. Run diagnostic UETP NIA20 test in user mode for at least four hours.

10. Disable the operating system.

11. Remove all field service packs and tapes from the customer's system, and store in a secure area.

12. Transfer/sign-off system to customer's authorized representative.

Digital Equipment Corporation • Marlboro, MA 01752